

# Towards a Visipedia: Combining Computer Vision and Communities of Experts

Thesis by  
Grant Van Horn

In Partial Fulfillment of the Requirements for the  
Degree of  
Doctor of Philosophy



CALIFORNIA INSTITUTE OF TECHNOLOGY  
Pasadena, California

2019  
Defended September 7, 2018

© 2019

Grant Van Horn

ORCID: 0000-0003-2953-9651

All rights reserved

## ACKNOWLEDGEMENTS

First and foremost I would like to thank my parents, Mathew and Mary Van Horn. I dedicate this work to them.

I would like to thank my advisor, Pietro Perona. I hope that at least a fraction of his curiosity for the world has worn off on me. It has been a pleasure to be a Slacker in his lab.

I would like to thank Serge Belongie, who I consider a joint advisor. I owe most of my opportunities from the last decade to Serge, and owe my current path in life to his guidance. I am forever grateful that he advised me during my undergraduate career, my Masters, and my Ph.D.

I would like to thank Steve Branson, easily my most influential mentor. From Steve, I learned how to identify problems, how to conduct research, and how to discuss the findings. I owe a lot to him, and I am very proud of the work we accomplished together.

I would like to thank Jessie Barry and the rest of her team at the Cornell Lab of Ornithology. I would also like to thank Scott Loarie and the rest of the iNaturalist team. I am grateful that I could work on both the Merlin and iNaturalist applications during my graduate studies, helping me connect my passion for computer science with my passion for the outdoors.

Finally, I owe my most enjoyable moments at Caltech to the Vision Lab, and I would like to thank all the Slackers I overlapped with: David Hall, Oisín Mac Aodha, Matteo Ruggero Ronchi, Joe Marino, Ron Appel, Mason McGill, Sara Beery, Alvita Tran, Natalie Bernat, Eyrun Eyjolfssdottir, Bo Chen, Krzysztof Chałupka, Serim Ryou, Cristina Segalin, Eli Cole, Jennifer Sun, Jan Dirk Wegner, Daniel Laumer, Michael Maire, Xavier Burgos-Artizzu, Conchi Fernandez, Louise Naud, Michele Damian, and Genevieve Patterson.

## ABSTRACT

Motivated by the idea of a Visipedia, where users can search and explore by image, this thesis presents tools and techniques for empowering expert communities through computer vision. The collective aim of this work is to provide a scalable foundation upon which an application like Visipedia can be built. We conduct experiments using two highly motivated communities, the birding community and the naturalist community, and report results and lessons on how to build the necessary components of a Visipedia. First, we conduct experiments analyzing the behavior of state-of-the-art computer vision classifiers on long tailed datasets. We find poor feature sharing between classes, potentially limiting the applicability of these models and emphasizing the ability to intelligently direct data collection resources. Second, we devise online crowdsourcing algorithms to make dataset collection for binary labels, multi-class labels, keypoints, and multi-instance bounding boxes faster, cheaper, and more accurate. These methods jointly estimate labels, worker skills, and train computer vision models for these tasks. Experiments show that we can achieve significant cost savings compared to traditional data collection techniques, and that we can produce a more accurate dataset compared to traditional data collection techniques. Third, we present two fine-grained datasets, detail how they were constructed, and analyze the test accuracy of state-of-the-art methods. These datasets are then used to create applications that help users identify species in their photographs: Merlin, an app assisting users in identifying birds species, and iNaturalist, an app that assists users in identifying a broad variety of species. Finally, we present work aimed at reducing the computational burden of large scale classification with the goal of creating an application that allows users to classify tens of thousands of species in real time on their mobile device. As a whole, the lessons learned and the techniques presented in this thesis bring us closer to the realization of a Visipedia.



## PUBLISHED CONTENT AND CONTRIBUTIONS

Van Horn, Grant and Pietro Perona (2019). “Reducing Memory & Computation Demands for Large Scale Visual Classification”.

G.V.H. participated in designing the project, developing the method, running the experiments and writing the manuscript.

Van Horn, Grant, Steve Branson, Scott Loarie, et al. (2018). “Lean Multiclass Crowdsourcing”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT. DOI: 10.1109/cvpr.2018.00287.

G.V.H. participated in designing the project, developing the method, running the experiments and writing the manuscript.

Van Horn, Grant, Oisin Mac Aodha, et al. (2018). “The iNaturalist Species Classification and Detection Dataset”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT. DOI: 10.1109/CVPR.2018.00914.

G.V.H. participated in designing the project, developing the method, running the experiments and writing the manuscript.

Branson, Steve, Grant Van Horn, and Pietro Perona (2017). “Lean Crowdsourcing: Combining Humans and Machines in an Online System”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7474–7483. DOI: 10.1109/CVPR.2017.647.

G.V.H. participated in designing the project, developing the method, running the experiments and writing the manuscript.

Van Horn, Grant and Pietro Perona (2017). “The Devil is in the Tails: Fine-grained Classification in the Wild”. In: *arXiv preprint arXiv:1709.01450*. URL: <https://arxiv.org/abs/1709.01450>.

G.V.H. participated in designing the project, developing the method, running the experiments and writing the manuscript.

Van Horn, Grant, Steve Branson, Ryan Farrell, et al. (2015). “Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604. DOI: 10.1109/CVPR.2015.7298658.

G.V.H. participated in designing the project, developing the method, running the experiments and writing the manuscript.

# TABLE OF CONTENTS

Acknowledgements . . . . .	iii
Abstract . . . . .	iv
Published Content and Contributions . . . . .	v
Table of Contents . . . . .	vi
Chapter I: Introduction . . . . .	1
Chapter II: The Devil is in the Tails: Fine-grained Classification in the Wild . . . . .	7
2.1 Abstract . . . . .	7
2.2 Introduction . . . . .	7
2.3 Related Work . . . . .	9
2.4 Experiment Setup . . . . .	11
2.5 Experiments . . . . .	13
2.6 Discussion and Conclusions . . . . .	21
Chapter III: Lean Crowdsourcing: Combining Humans and Machines in an Online System . . . . .	29
3.1 Abstract . . . . .	29
3.2 Introduction . . . . .	29
3.3 Related Work . . . . .	31
3.4 Method . . . . .	33
3.5 Models For Common Types of Annotations . . . . .	38
3.6 Binary Annotation . . . . .	38
3.7 Part Keypoint Annotation . . . . .	40
3.8 Multi-Object Bounding Box Annotations . . . . .	42
3.9 Experiments . . . . .	47
3.10 Conclusion . . . . .	54
Chapter IV: Lean Multiclass Crowdsourcing . . . . .	60
4.1 Abstract . . . . .	60
4.2 Introduction . . . . .	60
4.3 Related Work . . . . .	62
4.4 Multiclass Online Crowdsourcing . . . . .	64
4.5 Taking Pixels into Account . . . . .	70
4.6 Experiments . . . . .	71
4.7 Conclusion . . . . .	76
Chapter V: Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection . . . . .	81
5.1 Abstract . . . . .	81
5.2 Introduction . . . . .	81
5.3 Related Work . . . . .	84
5.4 Crowdsourcing with Citizen Scientists . . . . .	86
5.5 NABirds . . . . .	88

5.6	Annotator Comparison . . . . .	89
5.7	Measuring the Quality of Existing Datasets . . . . .	91
5.8	Effect of Annotation Quality & Quantity . . . . .	93
5.9	Conclusion . . . . .	96
5.10	Acknowledgments . . . . .	97
Chapter VI: The iNaturalist Species Classification and Detection Dataset . . .		101
6.1	Abstract . . . . .	101
6.2	Introduction . . . . .	101
6.3	Related Datasets . . . . .	103
6.4	Dataset Overview . . . . .	104
6.5	Experiments . . . . .	110
6.6	Conclusions and Future Work . . . . .	118
Chapter VII: Reducing Memory & Computation Demands for Large Scale Visual Classification . . . . .		126
7.1	Abstract . . . . .	126
7.2	Introduction . . . . .	126
7.3	Related Work . . . . .	129
7.4	Taxonomic Parameter Sharing . . . . .	131
7.5	Experiments . . . . .	134
7.6	Conclusion . . . . .	141
Chapter VIII: Conclusions and Future Directions . . . . .		147

## Chapter 1

### INTRODUCTION

Visipedia, a community-generated visual encyclopedia, is the primary motivator and inspiration for the work in this thesis. The Visipedia project<sup>1</sup> has been spearheaded by Pietro Perona’s group at Caltech and Serge Belongie’s group, first at UCSD and then Cornell Tech. This thesis is the most recent in a series of theses (Welinder, 2012; Branson, 2012; Wah, 2014), coming out of Perona and Belongie’s respective groups, that attempts to make Visipedia a reality. In (Perona, 2010), Perona specifies the vision for Visipedia, defines the users and challenges of such a system, and muses on its feasibility. He identifies two primary interfaces that Visipedia must provide.

First, Visipedia must provide an interface that allows users to ask visual questions. Perona imagines an interface that can segment a photograph into its meaningful component regions and then associates each of those regions with their corresponding Wikipedia entry or to the same region in vast collections of photographs. This would enable a user to photograph a rock pigeon and then click on the *operculum* (i.e. the white, fleshy part at the base of the bill) to learn more about what purpose that structure serves. Similarly, this type of interface would enable a user to navigate to the Wikipedia page for *Amanita pantherina* simply from a photograph of that fungus. Similar types of interactions could be had with photographs outside the natural world: a photograph of a painting could be annotated with the artist’s information; a photograph of a car engine could be annotated with the engine part names and replacement information; a photograph of a retina could be annotated with defects and linked to similar clinical cases.

To provide answers to the visual queries discussed in the previous paragraph, Visipedia must first be made aware of the visual properties of the world and their relationships. This is the second primary interaction with Visipedia: an interface that allows experts to share their visual knowledge of the world. Perona imagines an easy-to-use annotation interface that allows experts to contribute their visual knowledge by annotating a few paradigmatic images. An ornithologist could provide information on bird morphology for a few different families. A mycologist could provide example photographs of fungi species. A Chevrolet engineer could

---

<sup>1</sup>[www.visipedia.org](http://www.visipedia.org)

provide engine part schematics. An ophthalmologist could provide example retina images along with their prognosis. Perona emphasizes the importance of making this interface easy and quick, as experts' time is scarce and valuable, and annotating all of the important regions of an image is laborious and boring.

A high degree of automation is required to power the interactions that make these two interfaces useful. A user, with a fleeting curiosity to identify the white, fleshy bit of a pigeon, would prefer immediate results rather than waiting for a human expert to answer. Similarly, an ornithologist would not annotate thousands of images with the anatomical parts of a bird. Instead, we would prefer if machines could analyze images and immediately return results and efficiently propagate information from tens of examples to thousands or millions of photographs. This introduces two additional types of people that would interact with Visipedia: annotators and machine vision researchers.

Annotators, or “eye balls for hire” (Perona, 2010), are the bridge connecting the few samples provided by an expert to the thousands of examples required to train modern computer vision models. Annotators could be paid crowd workers (e.g. Amazon Mechanical Turk workers), they could be motivated enthusiasts (e.g. citizen scientists) or they could be people tasked with doing a few annotations while trying to achieve another goal (e.g. GWAPs (Von Ahn and Dabbish, 2004) or Captchas (Von Ahn, Maurer, et al., 2008)). In any case, their job is to propagate the expert information to additional training data that can be used to train a computer vision model to do the task. Machine vision researchers are responsible for designing and implementing these computer vision models. These models are then responsible for annotating an image with hyperlinks that allow users to answer visual questions (i.e. clickable component regions), working with experts to efficiently incorporate their visual knowledge, and propagating expert information to additional images (effectively annotating the images of the web).

At this point, we have defined Visipedia as a community-generated visual encyclopedia that has interfaces to answer visual questions and that enable experts to share their visual knowledge. Annotators help propagate expert knowledge to additional images, producing datasets that can be used to train computer vision models designed by machine vision researchers. These same models power the question-answering interface and interact with experts to efficiently incorporate their knowledge. In (Perona, 2010), Perona discusses the challenges of actually building Visipedia from the perspective of a computer vision researcher in 2009. He noted that computer

vision models at the time were not capable of performing at the level of accuracy necessary to be useful, and that the field had not yet attempted to build such complex, heterogeneous systems. In addition, he observed that self-diagnosing models (capable of deciding when they should ask questions of humans), active incremental learning (necessary for learning in the large scale, dynamic web environment), and human-machine interaction were research topics largely ignored by the computer vision research community, yet crucial for Visipedia. It has been 9 years since Perona penned his vision, where do we stand now?

Powered by the return of convolutional neural networks (Krizhevsky, Sutskever, and Hinton, 2012), hardware advancements and easy-to-use computational libraries (Martin Abadi et al., 2015), the computer vision field as a whole has made incredible progress during my graduate studies on the tasks of image classification (Krizhevsky, Sutskever, and Hinton, 2012), object detection (Ren et al., 2017), keypoint localization (Chen et al., 2018), and image segmentation (He et al., 2017). Indeed, setting aside the feasibility of collecting training data, the costs of training, and the size of the resulting model, if a sufficiently large dataset can be collected for one the previous tasks, then often the performance of the resulting convolutional neural network model is adequate for production usage. Evidence of this progress can be seen in the availability of computer vision-powered applications now available to consumers. During my graduate career, I helped build two of these applications (iNaturalist and Merlin), available through the Google Play Store and Apple App Store, that help users identify species in their photographs. The iNaturalist app<sup>2</sup> has a server-based computer vision classifier that can help identify 25,000 species. The Merlin app<sup>3</sup> has a computer vision classifier available directly on the phone and can help users classify 2,000 bird species. Besides mobile applications, perhaps the most impressive sign of progress is the availability of self-driving cars (albeit limited in their scope for now) becoming available to consumers.

Computer vision has entered an era of big data, where the ability to collect larger datasets – larger in terms of the number of classes, the number of images per class, and the level of annotation per image – appears to be paramount for continuing performance improvement and expanding the set of solvable applications. However, while the accuracy of computer vision models has seen a rapid improvement over the last half-decade, our ability to collect datasets of sufficient size to train and evaluate these models has remained essentially unchanged and presents a significant hurdle

---

<sup>2</sup><https://www.inaturalist.org>

<sup>3</sup><http://merlin.allaboutbirds.org/>

to expanding the availability and utility of computer vision services. Indeed, the title of this thesis is “Towards a Visipedia,” not “Visipedia: Mission Accomplished.” So in the interim period between (Perona, 2010) and this thesis, many of the key challenges to actually building a Visipedia (namely the challenges associated with annotating large datasets) were still largely ignored (excluding the contributions of the previous theses on Visipedia). The work in this thesis, however, is aimed at reducing the burden of collecting datasets and will hopefully lay the foundation for building a Visipedia.

In (Perona, 2010), Perona suggested that a step towards integrating a Visipedia with all of Wikipedia would be to focus on a well-defined domain with a community of highly motivated enthusiasts. This is precisely what we have done by engaging with the birding community through the Cornell Lab of Ornithology and the naturalist community through iNaturalist. The following chapters, each of which is self-contained, explore dataset properties, efficient methods of collection, and training state-of-the-art methods for deploying classification services to these two communities. In terms of building a broader Visipedia, the following chapters contain useful information for interacting with different types of annotators, modeling the skills of annotators and vision models, and how to reliably combine information from multiple sources (both human and machine). Taken as whole, this thesis is an attempt to fill in the missing pieces that provide the required automation to make Visipedia a reality. I will briefly summarize the chapters and the relevant contributions.

In Chapter 1, we discuss the long tail property of real world datasets and the effect this tail has on classification performance. Experiments show that state-of-the-art methods do not share feature learning between classes and that new training methodologies or collecting additional data in the tail is required to improve performance.

In Chapter 2, we devise a method for online crowdsourcing of binary labels, keypoints, and multi-instance bounding box annotations. This method is capable of estimating worker skills and jointly trains computer vision models. We present experiments that show significant cost savings and improvements in dataset accuracy by using our model instead of traditional dataset collection techniques.

In Chapter 3, we extend our online crowdsourcing method to large-scale multiclass annotations. Our method is capable of utilizing a taxonomy across the labels, handling a dependence between the annotations, and jointly training a computer

vision system. We present experiments that show significant accuracy gains over traditional majority vote techniques.

In Chapter 4, we present the NABirds dataset, collected by the birding community through the Cornell Lab of Ornithology. We present experiments comparing the annotation performance of different groups of workers on different types of tasks. We describe the benefit of tapping into a motivated community and how to best harness its enthusiasm. We additionally present results on dataset noise and show that modern state-of-the-art methods are resilient to a reasonable amount of noise.

In Chapter 5, we present the iNaturalist Species Classification and Detection Dataset, collected by the naturalist community through iNaturalist. We describe dataset collection and prepping methods and evaluate state-of-the-art classifiers and detectors. In addition, we conduct a competition to motivate the computer vision research community to explore large-scale, fine-grained classification and detection.

In Chapter 6, we analyze multiple techniques for reducing the computational burden of the final fully connected layer of traditional convolutional networks. We experiment with a novel taxonomic approach but find that a simple factorization and training scheme allows us to reduce the amount of computation and memory by 25x without any loss in accuracy.

Finally, in Chapter 7, I suggest directions for future work.

## References

- Branson, Steven (2012). “Interactive learning and prediction algorithms for computer vision applications”. PhD thesis. UC San Diego.
- Chen, Yilun et al. (2018). “Cascaded pyramid network for multi-person pose estimation”. In: *CVPR*.
- He, Kaiming et al. (2017). “Mask r-cnn”. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, pp. 2980–2988.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks.” In: *NIPS*.
- Martin Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: <http://tensorflow.org/>.
- Perona, Pietro (2010). “Vision of a Visipedia”. In: *Proceedings of the IEEE* 98.8, pp. 1526–1534.



- Ren, Shaoqing et al. (2017). “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *PAMI*.
- Von Ahn, Luis and Laura Dabbish (2004). “Labeling images with a computer game”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 319–326.
- Von Ahn, Luis, Benjamin Maurer, et al. (2008). “recaptcha: Human-based character recognition via web security measures”. In: *Science* 321.5895, pp. 1465–1468.
- Wah, Catherine Lih-Lian (2014). “Leveraging Human Perception and Computer Vision Algorithms for Interactive Fine-Grained Visual Categorization”. PhD thesis. UC San Diego.
- Welinder, Nils Peter Egon (2012). “Hybrid human-machine vision systems : image annotation using crowds, experts and machines”. PhD thesis. California Institute of Technology.

## Chapter 2

# THE DEVIL IS IN THE TAILS: FINE-GRAINED CLASSIFICATION IN THE WILD

Van Horn, Grant and Pietro Perona (2017). “The Devil is in the Tails: Fine-grained Classification in the Wild”. In: *arXiv preprint arXiv:1709.01450*. URL: <https://arxiv.org/abs/1709.01450>.

## 2.1 Abstract

The world is long-tailed. What does this mean for computer vision and visual recognition? The main two implications are: (1) the number of categories we need to consider in applications can be very large, and (2) the number of training examples for most categories can be very small. Current visual recognition algorithms have achieved excellent classification accuracy. However, they require many training examples to reach peak performance, which suggests that long-tailed distributions will not be dealt with well. We analyze this question in the context of eBird, a large fine-grained classification dataset and a state-of-the-art deep network classification algorithm. We find that: (a) peak classification performance on well-represented categories is excellent, (b) given enough data, classification performance suffers only minimally from an increase in the number of classes, (c) classification performance decays precipitously as the number of training examples decreases, and (d) surprisingly, transfer learning is virtually absent in current methods. Our findings suggest that our community should come to grips with the question of long tails.

## 2.2 Introduction

During the past five years we have witnessed dramatic improvement in the performance of visual recognition algorithms (Russakovsky et al., 2015). Human performance has been approached or achieved in many instances. Three concurrent developments have enabled such progress: (a) the invention of ‘deep network’ algorithms where visual computation is learned from the data rather than hand-crafted by experts (Fukushima and Miyake, 1982; LeCun et al., 1989; Krizhevsky, Sutskever, and Hinton, 2012), (b) the design and construction of large and well-annotated datasets (Fei-Fei, Fergus, and Perona, 2004; Everingham and al., 2005; Deng et al., 2009; Tsung-Yi Lin et al., 2014) supplying researchers with a sufficient amount of

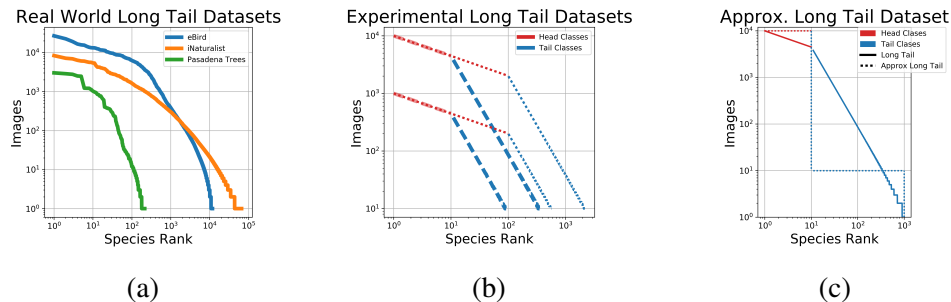


Figure 2.1: **(a)** The world is long-tailed. Class frequency statistics in real world datasets (birds, a wide array of natural species, and trees). These are long-tailed distributions where a few classes have many examples and most classes have few. **(b)** The 4 experimental long tail datasets used in this work. We modeled the eBird dataset (blue curve in (a)) and created four long tail datasets by shifting the modeled eBird dataset down (fewer images) and to the left (fewer species) by different amounts. Classes are split into head and tail groups; images per class in the respective groups decay exponentially. **(c)** Approximation of a long tail dataset. This approximation allows us to more easily study the effects of head classes on tail class performance.

data to train learning-based algorithms, and (c) the availability of inexpensive and ever more powerful computers, such as GPUs (Lindholm et al., 2008), for algorithm training.

Large annotated datasets yield two additional benefits, besides supplying deep nets with sufficient training fodder. The first is offering common performance benchmarks that allow researchers to compare results and quickly evolve the best algorithmic architectures. The second, more subtle but no less important, is providing researchers with a compass – a definition of the visual tasks that one ought to try and solve. Each new dataset pushes us a bit further towards solving real world challenges. We wish to focus here on the latter aspect.

One goal of visual recognition is to enable machines to recognize objects in the world. What does the world look like? In order to better understand the nature of visual categorization in the wild we examined three real-world datasets: bird species, as photographed worldwide by birders who are members of eBird (Sullivan et al., 2009); tree species, as observed along the streets of Pasadena (Wegner et al., 2016); and plants and animal species, as photographed by the iNaturalist ([www.inaturalist.org](http://www.inaturalist.org)) community. One salient aspect of these datasets is that some species are very frequent, while most species are represented by only few specimens (Fig 2.1a). In a nutshell: the world is long-tailed, as previously noted in the context of subcategories and object views (Salakhutdinov, Torralba, and Tenenbaum,

2011; Zhu, Anguelov, and Ramanan, 2014). This is in stark contrast with current datasets for visual classification, where specimen distribution per category is almost uniformly distributed (see (Tsung-Yi Lin et al., 2014) Figure 5(a)).

With this observation in mind, we ask whether current state-of-the-art classification algorithms, whose development is motivated and benchmarked by uniformly distributed datasets, deal well with the world’s long tails. Humans appear to be able to generalize from few examples; can our algorithms do the same? Our experiments show that the answer is *no*. While, when data is abundant, machine-vision classification performance can currently rival humans, we find that this is emphatically not the case when data is scarce for most classes, even if a few are abundant.

This work is organized as follows: In Section 2.3, we review the related work. We then describe the datasets and training process in Section 2.4, followed by an analysis of the experiments in Section 2.5. We summarize and conclude in Section 2.6.

## 2.3 Related Work

**Fine-Grained Visual Classification** – The vision community has released many fine-grained datasets covering several domains such as birds (Welinder et al., 2010; Wah et al., 2011; Berg, Liu, et al., 2014; Van Horn et al., 2015), dogs (Khosla et al., 2011; Liu et al., 2012), airplanes (Maji et al., 2013; Vedaldi et al., 2014), flowers (Nilsback and Zisserman, 2006), leaves (Kumar et al., 2012), trees (Wegner et al., 2016) and cars (Krause, Stark, et al., 2013; Y.-L. Lin et al., 2014). These datasets were constructed to be uniform, or to contain "enough" data for the task. The recent Pasadena Trees dataset (Wegner et al., 2016) is the exception. Most fine-grained research papers present a novel model for classification (Xu et al., 2015; Tsung-Yu Lin, RoyChowdhury, and Maji, 2015; Farrell et al., 2011; Krause, Jin, et al., 2015; Xie et al., 2015; Branson et al., 2014; Gavves et al., 2015; Simon and Rodner, 2015; Göring et al., 2014; Shih et al., 2015; N. Zhang et al., 2014; Berg and Belhumeur, 2013; Chai, Lempitsky, and Zisserman, 2013; Xiao et al., 2015; Y. Zhang et al., 2016; Pu et al., 2014). While these methods often achieve state-of-the-art performance at the time of being published, it is often the case that the next generation of convolutional networks can attain the same level of performance without any custom modifications. In this work, we use the Inception-v3 model (Szegedy et al., 2016), pretrained on ImageNet for our experiments. Some of the recent fine-grained papers have investigated augmenting the original datasets with additional data from the web (Krause, Sapp, et al., 2016; Xu et al., 2015;

Xie et al., 2015; Van Horn et al., 2015). Krause et al. (Krause, Sapp, et al., 2016) investigated the collection and use of a large, noisy dataset for the task of fine-grained classification and found that off the shelf CNN models can readily take advantage of these datasets to increase accuracy and reach state-of-the-art performance. Krause et al. mention, but do not investigate, the role of the long tail distribution of training images. In this work, we specifically investigate the effect of this long tail on the model performance.

**Imbalanced Datasets** – Techniques to handle imbalanced datasets are typically split into two regimes: algorithmic solutions and data solutions. In the first regime, cost-sensitive learning (Elkan, 2001) is employed to force the model to adjust its decision boundaries by incurring a non-uniform cost per misclassification; see (H. He and Garcia, 2009) for a review of the techniques. The second regime concerns data augmentation, achieved either through over-sampling the minority classes, under sampling the majority classes, or synthetically generating new examples for the minority classes. When using mini batch gradient descent (as we do in the experiments), oversampling the minority classes is similar to weighting these classes more than the majority classes, as in cost-sensitive learning. We conduct experiments on over-sampling the minority classes. We also employ affine (Krizhevsky, Sutskever, and Hinton, 2012) and photometric (Howard, 2013) transformations to synthetically boost the number of training examples.

**Transfer Learning** – Transfer learning (Pan and Yang, 2010) attempts to adapt the representations learned in one domain to another. In the era of deep networks, the simplest form of transfer learning is using features extracted from pretrained ImageNet (Russakovsky et al., 2015) or Places (Zhou et al., 2014) networks, see (Sharif Razavian et al., 2014; Donahue et al., 2014). The next step is actually fine-tuning (Girshick et al., 2014) these pretrained networks for the target task (Yosinski et al., 2014; Agrawal, Girshick, and Malik, 2014; Oquab et al., 2014; Huh, Agrawal, and Efros, 2016). This has become the standard method for obtaining baseline numbers on a new target dataset and often leads to impressive results (Azizpour et al., 2015), especially when the target dataset has sufficient training examples. More sophisticated transfer learning methods (Long et al., 2015; Tzeng et al., 2015) are aimed at solving the domain adaptation problem. In this work, we are specifically interested in a single domain, which happens to contain a long tail distribution of training data for each class. We investigate whether there is a transfer of knowledge from the well represented classes to the sparsely represented classes.

**Low Shot Learning** – We experiment with a minimum of 10 training images per class, which falls into the realm of low shot learning, a field concerned with learning novel concepts from few examples. In (Wang and Hebert, 2016b), Wang and Hebert learn a regression function from classifiers trained on small datasets to classifiers trained on large datasets, using a fixed feature representation. Our setup is different in that we want to allow our feature representation to adapt to the target dataset, and we want a model that can classify both the well represented classes and the sparsely represented classes. The recent work of Hariharan and Girshick in (Hariharan and Girshick, n.d.) explored this setup specifically, albeit in the broad domain of ImageNet. The authors propose a low shot learning benchmark and implement a loss function and feature synthesis scheme to boost performance on under represented classes. However, their results showed marginal improvement when using a high capacity model (at 10 images per class the ResNet-50 (K. He et al., 2016) model performed nearly as well as their proposed method). Our work aims to study the relationship between the well represented classes and the sparse classes, within a single domain. Metric learning tackles the low-shot learning problem by learning a representation space where distance corresponds to similarity. While these techniques appear promising and provide benefits beyond classification, they do not hold up well against simple baseline networks for the specific task of classification (Rippel et al., 2015).

## 2.4 Experiment Setup

### Datasets

We consider three different types of datasets: uniform, long tail, and approximate long tail. We used images from eBird (ebird.org) to construct each of these datasets. These images are real world observations of birds captured by citizen scientists and curated by regional experts. Each dataset consists of a training, validation, and test split. When placing images into each split, we ensure that a photographer’s images do not end up in multiple splits for a single species. The test set is constructed to contain as many different photographers as possible (e.g. 30 images from 30 different photographers). The validation set is similarly constructed, and the train set is constructed from the remaining photographers.

**Uniform Datasets** – The uniform datasets allow us to study the performance of the classification model under optimal image distribution conditions. These datasets have the same number of images per class: either 10, 100, 1K, or 10K. The total number of classes can be either 10, 100, or 1K. We did not analyze a uniform

dataset with 1K classes containing 1K or 10K images each due to a lack of data from eBird. Each smaller dataset is completely contained within the larger dataset (e.g. the 10 class datasets are contained within the 100 class datasets, etc.). The test and validation sets are uniform, with 30 and 10 images for each class respectively, and remain fixed for a class across all uniform datasets.

**Approx. Long Tail Datasets** – To conveniently explore the effect of moving from a uniform dataset to a long tail dataset we constructed approximate long tail datasets, see Figure 2.1c. These datasets consist of 1K classes split into two groups: the head classes and the tail classes. All classes within a group have the same number of images. We study two different sized splits: a 10 head, 990 tail split and a 100 head, 900 tail split. The 10 head split can have 10, 100, 1K, or 10K images in each head class. The 100 head split can have 10, 100, or 1K images in each head class. The tail classes from both splits can have 10 or 100 images. We use the validation and test sets from the 1K class uniform dataset for all of the approximate long tail datasets. This allows us to compare the performance characteristics of the different datasets in a reliable way, and we can use the 1K class uniform datasets as reference points.

**Long Tail Datasets** – The full eBird dataset, with nearly 3 million images, is not amenable to easy experimentation. Rather than training on the full dataset, we would prefer to model the image distribution and use it to construct smaller, tunable datasets, see Figure 2.1b. We did this by fitting a two-piece broken power law to the eBird image distribution. Each class,  $i \in [1, N]$ , is put into the head group if  $i \leq h$ , otherwise it is put into the tail group, where  $h$  is the number of head classes. Each head class  $i$  contains  $y \cdot i^{a_1}$  images, where  $y$  is the number of images in the most populous class and  $a_1$  is the power law exponent for the head classes. Each tail class  $i$  has  $y \cdot h^{(a_1-a_2)} \cdot i^{a_2}$  where  $a_2$  is the power law exponent for the tail classes. We used linear regression to determine that  $a_1 = -0.3472$  and  $a_2 = -1.7135$ . We fixed the minimum number of images for a class to be 10. This leaves us with 2 parameters that we can vary:  $y$ , which shifts the distribution up and down, and  $h$  which shifts the distribution left and right. We analyze four long tail datasets by selecting  $y$  from  $\{1K, 10K\}$  and  $h$  from  $\{10, 100\}$ . Each resulting dataset consists of a different number of classes and therefore has a different test and validation split. We keep to the pattern of reserving 30 test images and 10 validation images for each class.

### **Model Training & Testing Details**

**Model** – We use the Inception-v3 network (Szegedy et al., 2016), pretrained from ILSVC 2012, as the starting point for all experiments. The Inception-v3 model exhibits good trade-off between size of the model (27.1M parameters) and classification accuracy on the ILSVC (78% top 1 accuracy) as compared to architectures like AlexNet and VGG. We could have used the ResNet-50 model but opted for Inception-v3, as it is currently being used by the eBird development team.

**Training** – We have a fixed training regime for each dataset. We fine-tune the pretrained Inception-v3 network (using TensorFlow (Martin Abadi et al., 2015)) by training all layers using a batch size of 32. Unless noted otherwise, batches are constructed by randomly sampling from the pool of all training images. The initial learning rate is 0.0045 and is decayed exponentially by a factor of 0.94 every 4 epochs. Training augmentation consists of taking random crops from the image whose area can range from 10% to 100% of the image, and whose aspect ratio can range from 0.7 to 1.33. The crop is randomly flipped and has random brightness and saturation distortions applied.

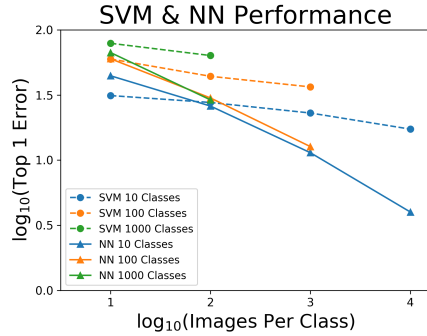
**Testing** – We use the validation loss to stop the training by waiting for it to steadily increase, signaling that the model is overfitting. We then consider all models up to this stopping point and use the model with the highest validation accuracy for testing. At test time, we take a center crop of the image, covering 87.5% of the image area. We track top 1 image accuracy as the metric, as is typically used in fine-grained classification benchmarks. Note that image accuracy is the same as class average accuracy for datasets with uniform validation and test sets, as is the case for all of our experiments.

## **2.5 Experiments**

### **Uniform Datasets**

We first study the performance characteristics of the uniform datasets. We consider two regimes: (1) we extract feature vectors from the pretrained network and train a linear SVM; and (2) we fine-tune the pretrained network, see Section 2.4 for the training protocol. We use the activations of the layer before the final fully connected layer as our features for the SVM and used the validation set to tune the penalty parameter. Figure 2.2a plots the error achieved under these two different regimes. We can see that fine-tuning the neural network is beneficial in all cases except the extreme case of 10 classes with 10 images each (in which case the model overfit





(a)



(b)

**Figure 2.2: (a) Classification performance as a function of training set size on uniform datasets.** A neural network (solid lines) achieves excellent accuracy on these uniform datasets. Performance keeps improving as the number of training examples increases to 10K per class – each 10x increase in dataset size is rewarded with a 2x cut in the error rate. We also see that the neural net scales extremely well with increased number of classes, increasing error only marginally when 10x more classes are used. Neural net performance is also compared with SVM (dashed lines) trained on extracted ImageNet features. We see that fine-tuning the neural network is beneficial in all cases except in the extreme case of 10 classes with 10 images each. **(b) Example misclassifications.** Four of the twelve images misclassified by the 10 class, 10K images per class model. Clockwise from top left: Osprey misclassified as Great Blue Heron, Bald Eagle (center of image) misclassified as Great Blue Heron, Cooper’s Hawk misclassified as Great Egret, and Ring-billed Gull misclassified as Great Egret.

quickly, even with extensive hyperparameter sweeps). The neural network scales incredibly well with increasing number of classes, incurring a small increase in error for 10x increase in the number of classes. This should be expected given that the network was designed for 1000-way ImageNet classification. At 10k images per class, the network is achieving 96% accuracy on 10 bird species, showing that the network can achieve high performance given enough data. For the network, a 10x increase in data corresponds to at least a 2x error reduction. Keep in mind that the opposite is true as well: as we remove 10x images, the error rate increases by at least 2x. These uniform dataset results will be used as reference points for the long tail experiments.

### Uniform vs. Natural Sampling

The long tail datasets present an interesting question when it comes to creating the training batches: should we construct batches of images such that they are sampled

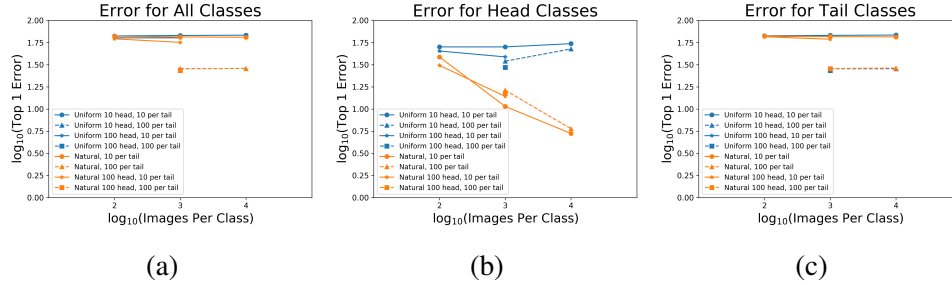
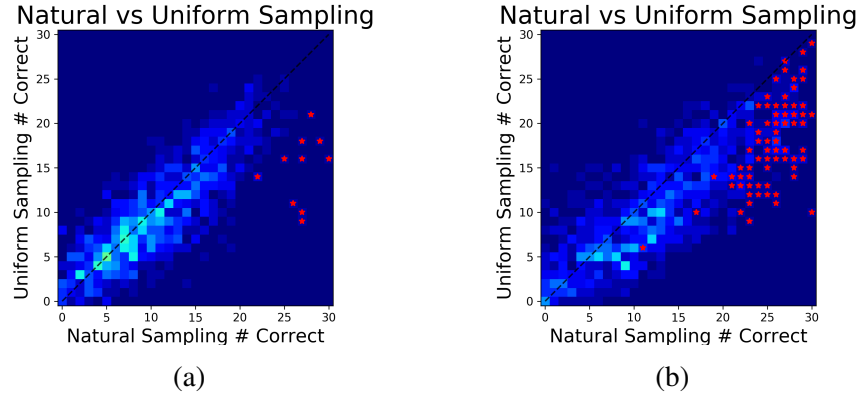


Figure 2.3: **Uniform vs. Natural Sampling – effect on error.** Error plots for models trained with uniform sampling and natural sampling. (a) The overall error of both methods is roughly equivalent, with natural sampling tending to be as good or better than uniform sampling. (b) Head classes clearly benefit from natural sampling. (c) Tail classes tend to have the same error under both sampling regimes.

uniformly from all classes, or such that they are sampled from the natural distribution of images? Uniformly sampling from the classes will result in a given tail image appearing more frequently in the training batches than a given head image, i.e., we are oversampling the tail classes. To answer this question, we trained a model for each of our approximate long tail datasets using both sampling methods and compared the results. Figure 2.3 plots the error achieved with the different sampling techniques on three different splits of the classes (all classes, the head classes, and the tail classes). We see that both sampling methods often converge to the same error, but the model trained with natural sampling is typically as good as or better than the model trained with uniform sampling. Figure 2.4 visualizes the performance of the classes under the two different sampling techniques for two different long tail datasets. These figures highlight that the head classes clearly benefit from natural sampling, and the center of mass of the tail classes is skewed slightly towards the natural sampling. The results for the long tail dataset experiments in the following sections use natural sampling.

### Transferring Knowledge from the Head to the Tail

Section 2.5 showed that the Inception-v3 architecture does extremely well on uniform datasets, achieving 96% accuracy on the 10 class, 10K images per class dataset; 87.3% accuracy on the 100 class, 1K images per class dataset; and 71.5% accuracy on the 1K class, 100 images per class dataset. The question we seek to answer is: how is performance affected when we move to a long tail dataset? Figure 2.5a summarizes our findings for the approximate long tail datasets (see Table 2.1 and Table 2.2 for the specific performance data). Starting with a dataset of 1000 classes



**Figure 2.4: Uniform vs. Natural Sampling – effect on accuracy.** We compare the effect of uniformly sampling from classes vs. sampling from their natural image distribution when creating training batches for long tailed datasets, Section 2.5. We use 30 test images per class, so correct classification rate is binned into 31 bins. It is clear that the head classes (marked as stars) benefit from the natural sampling in both datasets. The tail classes in **(a)** have an average accuracy of 32.1% and 34.2% for uniform and natural sampling respectively. The tail classes in **(b)** have an average accuracy of 33.5% and 38.6% for uniform and natural sampling respectively. For both plots, head classes have 1000 images and tail classes have 10 images.

and 10 images in each class, the top 1 accuracy across all classes is 33.2% (this is the bottom, leftmost blue point in the figure). If we designate 10 of the classes as head classes, and 990 classes as tail classes, what happens when we increase the number of available training data in the head classes (traversing the bottom blue line in Figure 2.5a)? We see that the head class accuracy approaches the peak 10 class performance of 96% accuracy (reaching 94.7%), while the tail classes have remained near their initial performance of 33.2%.

We see a similar phenomenon even if we are more optimistic regarding the number of available training images in the tail classes, using 100 rather than 10 (the top blue line in Figure 2.5a). The starting accuracy across all 1000 classes, each with 100 training images, is 71.5%. As additional images are added to the head classes, the accuracy on the head classes again approaches the peak 10 class performance (reaching 94%) while the tail classes are stuck at 71%.

We can be optimistic in another way by moving more classes into the head, therefore making the tail smaller. We now specify 100 classes to be in the head, leaving 900 classes for the tail (the green points in 2.5a). We see a similar phenomenon even in this case, although we do see a slight improvement for the tail classes when the 100 head classes have 1k images each. These experiments reveal that there is very

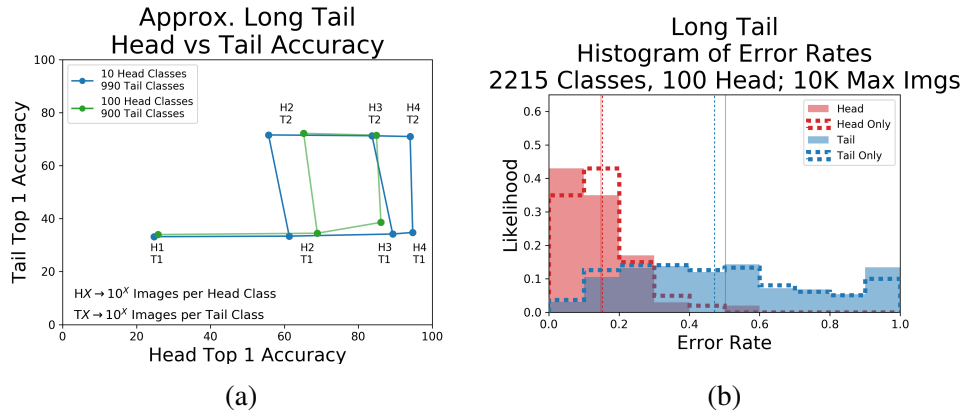
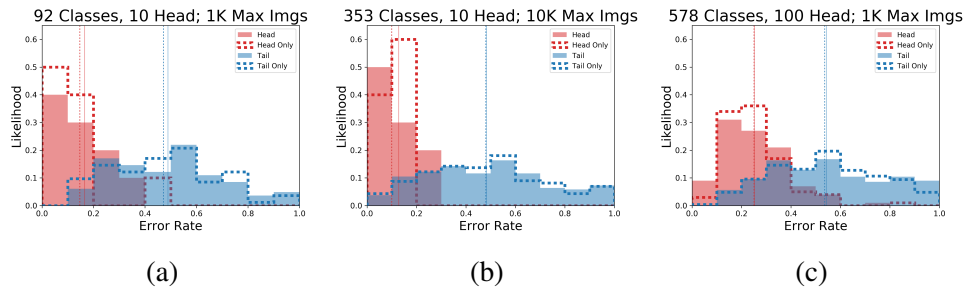


Figure 2.5: **Transfer between head and tail in approximate long tail datasets.**

(a) Head class accuracy is plotted against tail class accuracy as we vary the number of training examples in the head and in the tail for the approximate long tail datasets. Each point is associated with its nearest label. The labels indicate (in base 10) how much training data was in each head class (H) and each tail class (T). Lines between points indicate an increase in either images per head class, or images per tail class. As we increase images in the head class by factors of 10, the performance on the tail classes remains approximately constant. This means that there is a very poor transfer of knowledge from the head classes to the tail classes. As we increase the images per tail class, we see a slight loss in performance in the head classes. The overall accuracy of the model is vastly improved though. (b) **Histogram of error rates for a long tail dataset.** The same story applies here: the tail classes do not benefit from the head classes. The overall error of the joint head and tail model is 48.6%. See Figure 2.6 for additional details.

little to no transfer learning occurring within the network. Only the classes that receive additional training data actually see an improvement in performance, even though all classes come from the same domain. To put it plainly, an additional 10K bird images covering 10 bird species does nothing to help the network learn a better representation for the remaining bird species.

To confirm the results on the approximate long tail datasets, we experimented on four long tail distributions modeled after the actual eBird dataset, see Section 2.4 for details on the datasets. For these experiments, we trained three separate models, one trained with all classes and the other two trained with the head classes or tail classes respectively. Figures 2.5b and 2.6 show the results. We see the same recurring story: the tail performance is not affected by the head classes. Training a model solely on the tail classes is as good as, or even better, than training jointly with the head classes, even though the head classes are from the same domain and are doubling the size of the training dataset. The network is not transferring knowledge from the



**Figure 2.6: Histogram of Error Rates for Long Tail Datasets.** These plots compare the performance of the head and tail classes trained jointly (labeled Head and Tail respectively) vs. individually (labeled Head Only and Tail Only respectively). The dashed histograms represent the error rates for individual models (trained exclusively on the head (red) or tail (blue) classes), and the solid histograms represent the error rates of the head and tail classes within the joint model. The vertical lines mark the mean error rates. We see that the tail classes do not benefit from being trained with the head classes: the mean error rate of a model trained exclusively on the tail classes does as good or better than a model trained with both head and tail classes. The overall joint error of the models (dominated by the tail performance) are: 45.1% for (a), 47.1% for (b) and 49.2% for (c).

head classes to the tail classes. See Table 2.3 for the detailed results.

Dataset	Images / Head Class	Images / Tail Class	Overall ACC	Head ACC	Tail ACC
10 head classes 990 tail classes	100	100	71.5	55.7	71.6
	100	10	33.7	61.3	33.4
	1,000	10	34.8	89.3	34.2
	10,000	10	35.4	94.7	34.8
100 head classes 900 tail classes	100	100	71.5	65.2	72.2
	100	10	37.9	68.9	34.5
	1000	10	43.3	86.1	38.6

**Table 2.1: Top 1 accuracy for head and tail classes when going from uniform to approximate long tail image distribution.** The uniform dataset performance is the first row for the respective datasets; the subsequent rows are approximate long tail datasets. We see that the head classes benefit from the additional training images (Head ACC increases), but the tail classes benefit little, if at all (Tail ACC).

### Increasing Performance on the Head Classes

The experiments in Section 2.5 showed that we should not expect the tail classes to benefit from additional head class training data. While we would ultimately like to have a model that performs well on the head and tail classes, for the time being we may have to be content with optimizing for the classes that have sufficient training

Dataset	Images / Head	Images / Tail	Overall ACC	Head ACC	Tail ACC	Tail Isolated ACC	$\Delta$ Error Tail Isolated
10 H 990 T	10	10	33.2	24.7	33.2	33.4	-
	100	10	33.7	61.3	33.4	34.2	-1.2%
	1,000	10	34.8	89.3	34.2	36.4	-4.5%
	10,000	10	35.4	94.7	34.8	37.8	-6.6%
	100	100	71.5	55.7	71.6	71.8	-
	1,000	100	71.4	83.7	71.3	71.9	-0.4%
	10,000	100	71.3	94	71	72.6	-2.8%
100 H 900 T	10	10	33.2	25.8	34	35	-
	100	10	38	68.9	34.5	40.5	-8.5%
	1,000	10	43.4	86.1	38.6	50.9	-24.5%
	100	100	71.5	65.2	72.2	73.2	-
	1,000	100	72.8	84.9	71.5	75.3	-7.8%

Table 2.2: **Tail class performance.** This table details the tail class performance in uniform and approximate long tail datasets. In addition to showing the accuracy of the tail classes (Tail ACC) we show the performance of tail classes in isolation from the head classes (Tail Isolated ACC). To compute Tail Isolated ACC, we remove all head class images from the test set and ignore the head classes when making predictions on tail class images. These numbers reflect the situation of using the head classes to improve the feature representation of the network. The  $\Delta$  Error Tail Isolated column shows the decrease in error between the tail performance when the head classes are considered (Tail ACC) and the tail performance in isolation (Tail Isolated ACC). These numbers are a sanity check to ensure that the tail classes do indeed benefit from a feature representation learned with the additional head class images. The problem is that the benefit of the representation is not shared when both the head and tail classes are considered together.

data, i.e. the head classes. In this section, we explore whether we can use the tail to boost performance on the head classes. For each experiment, the model is trained on all classes specified in the training regime (which may be the head classes only, or could be the head and the tail classes), but at test time only head test images are used and only the head class predictions are considered (e.g. a model trained for 1000 way classification will be restricted to make predictions for the 10 head classes only).

We first analyze the performance of the head classes in a uniform dataset situation, where we train jointly with tail classes that have the same number of training images as the head classes. This can be considered the best case scenario for transfer learning as the source and target datasets are from the same distribution, and there

Dataset Params	Num Tail Classes	Overall ACC	Head ACC	Head Isolated ACC	Head Model ACC	Tail ACC	Tail Isolated ACC	Tail Model ACC
h = 10 y = 1K	82	54.9	85.7	88.7	87.7	51.2	56.8	53
h = 10 y = 10K	343	52.9	89.7	94.3	92.6	51.8	53.6	52.1
h = 100 y = 1K	478	50.8	76.5	79.7	76.5	45.4	53.6	46.1
h = 100 y = 10K	2115	51.4	87.4	89.4	87	49.7	53.4	53

Table 2.3: **Top 1 accuracy for the long tail datasets.** This table details the results of the long tail experiments. See Section 2.4 for information on the dataset parameters. Three different models were trained for each dataset. **1. Whole Model** This model was trained with both the head and the tail classes. Overall top 1 accuracy can be found in the Overall ACC column. Performance on the head and tail classes can be found in the Head ACC and Tail ACC columns respectively. Performance on the head and tail classes in isolation from each other can be found in the Head Isolated ACC and Tail Isolated ACC columns respectively. **2. Head Model** This model was trained exclusively on the head classes. Overall top 1 accuracy (on the head classes only) can be found in the Head Model ACC column. **3. Tail Model** This model was trained exclusively on the tail classes. Overall top 1 accuracy (on the tail classes only) can be found in the Tail Model ACC column.

are many more source classes than target classes. Figure 2.7a shows the results. In both the 10 head class situation and 100 head class situation, we see drops in error when jointly training the head and tail (dashed lines) as compared to the head only model (solid lines).

The next experiments explore the benefit to the head in the approximate long tail datasets. Figures 2.7b and 2.7c show the results. We found that there is a benefit to training with the long tail, between 6.3% and 32.5% error reduction, see Table 2.4. The benefit of the tail typically decreases as the ratio of head images to tail images increases. When this ratio exceeds 10, it is worse to use the tail during training.

In these experiments, we have been monitoring the performance of all classes during training with a uniform validation set (10 images per class). This validation set is our probe into the model, and we use it to select which iteration of the model to use for testing. We now know that using as much tail data as possible is beneficial. This raises the following question: can we monitor solely the head classes with the validation set and still recover an accurate model? If the answer is yes, then we will

be able to place all tail images in the training set rather than holding some out for the validation set. The results shown in Figure 2.8 show that this is possible, and that it actually produces a more accurate model for the head classes.

Dataset	Images / Head	Images / Tail	Head / Tail Image Ratio	Head Isolated ACC	Head Model ACC	$\Delta$ Error
10 H 990 T	10	10	0.01	<b>66.6</b>	55.6	-24.6%
	100	10	0.1	<b>77.3</b>	74	-12.7%
	1,000	10	1.01	<b>91.3</b>	88.6	-23.7%
	10,000	10	10.1	94.6	<b>96</b>	+35%
	100	100	0.01	<b>85.6</b>	74	-44.6%
	1,000	100	0.1	<b>92.3</b>	88.6	-32.5%
	10,000	100	1.01	<b>96.3</b>	96	-7.5%
100 H 900 T	10	10	0.11	<b>49</b>	40	-15%
	100	10	1.11	<b>74</b>	70.2	-12.8%
	1,000	10	11.11	86.8	<b>87.3</b>	+3.9%
	100	100	0.11	<b>81.6</b>	70.2	-38.3%
	1,000	100	1.11	<b>88.1</b>	87.3	-6.3%

Table 2.4: **Head class performance.** This table details the performance of the head classes under different training regimes. The Head Isolated ACC numbers show the top 1 accuracy on the head class images when using a model trained with both head and tail classes, but only makes predictions for the head classes at test time. The Head Model ACC numbers show the top 1 accuracy for a model that was trained exclusively on the head classes. We can see that it is beneficial to train with the tail classes until the head to tail image ratio exceeds 10, at which point it is better to train with the head classes only.

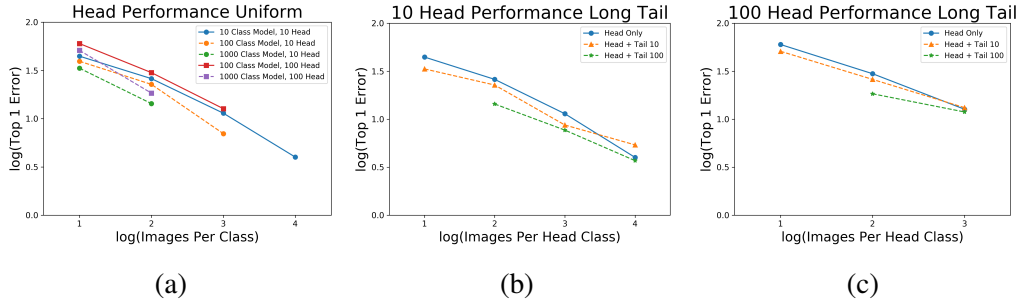
## 2.6 Discussion and Conclusions

The statistics of images in the real world are long-tailed: a few categories are highly represented, and most categories are observed only rarely. This is in stark contrast with the statistics of popular benchmark datasets, such as ImageNet (Deng et al., 2009), COCO (Tsung-Yi Lin et al., 2014), and CUB200 (Wah et al., 2011), where the training images are evenly distributed amongst classes.

We experimentally explored the performance of a state-of-the-art classification model on approximate and realistic long-tailed datasets. We make four observations which, we hope, will inform future research in visual classification.

First, performance is excellent, even in challenging tasks, when the number of training images exceeds many thousands. For example, the species classification





**Figure 2.7: Head class performance when using additional tail categories.** Head + Tail 10 refers to the tail having 10 images per class; Head + Tail 100 refers to the tail having 100 images per class. At test time we ignore tail class predictions for models trained with extra tail classes. We see that training with additional tail classes (dashed lines) decreases the error compared to a model trained exclusively on the head classes (solid lines) in both uniform and long tail datasets. In the long tail setting, the benefit is larger when the ratio of head images to tail images is smaller. We found that if this ratio exceeds 10, then it is better to train the model with the head classes only (right most points in (b) and (c)).

error rate is about 4% in the eBird dataset when each species is trained with  $10^4$  images (see Figure 2.2a). This is in line with the performance observed on ImageNet and COCO, where current algorithms can rival humans.

Second, if the number of training images is sufficient, classification performance suffers only minimally from an increase in the number of classes (see Figure 2.2a). This is indeed good news, as we estimate that there are tens of millions of object categories that one might eventually attempt to classify simultaneously.

Third, the number of training images is critical: classification error more than doubles every time we cut the number of training images by a factor of 10 (see Figure 2.2a). This is particularly important in a long-tailed regime since the tails contain most of the categories and therefore dominate average classification performance. For example: the largest long tail dataset from our experiments contains 550,692 images and yields an average classification error of 48.6% (see Figure 2.5b). If the same 550,692 images were distributed uniformly amongst the 2215 classes, the average error rate would be about 27% (see Fig. 2.2a). Another way to put it: collecting the eBird dataset took a few thousand motivated birders about 1 year. Increasing its size to the point that its top 2000 species contained at least  $10^4$  images would take 100 years (see Figure 2.1a). This is a long time to wait for excellent accuracy.

Fourth, on the datasets tested, transfer learning between classes is negligible with

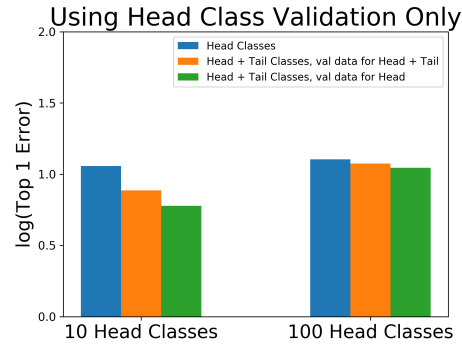


Figure 2.8: **Using validation data from the head classes only.** This plot shows the error achieved under different training regimes. **Head Classes** represents a model trained exclusively on the head classes, with 1000 training images each. The **Head + Tail Classes, val data for Head + Tail** represents a model trained with both head and tail classes (1000 images per head class, 100 images per tail class), and a validation set was used that had both head and tail class images. **Head + Tail Classes, val data for Head** represents a model trained with both head and tail classes (1000 images per head class, 100 images per tail class), and a validation set that only has head class images. We can see that it is beneficial to train with the extra tail classes, and that using the head classes exclusively in the validation set results in the best performing model.

current classification models. Simultaneously training on well-represented classes does little or nothing for the performance on those classes that are least represented. The average classification accuracy of the models will be dominated by the poor tail performance, and adding data to the head classes will not improve the situation.

Our findings highlight the importance of continued research in transfer and low shot learning (Fei-Fei, Fergus, and Perona, 2004; Hariharan and Girshick, n.d.; Wang and Hebert, 2016b; Wang and Hebert, 2016a) and provide baselines for future work to compare against. When we train on uniformly distributed datasets, we sweep the world’s long tails under the rug, and we do not make progress in addressing this challenge. As a community, we need to face up to the long-tailed challenge and start developing algorithms for image collections that mirror real-world statistics.

## References

Agrawal, Pulkrit, Ross Girshick, and Jitendra Malik (2014). “Analyzing the performance of multilayer neural networks for object recognition”. In: *European Conference on Computer Vision*. Springer, pp. 329–344.

- Azizpour, Hossein et al. (2015). “From generic to specific deep representations for visual recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 36–45.
- Berg, Thomas and Peter Belhumeur (2013). “POOF: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 955–962.
- Berg, Thomas, Jiongxin Liu, et al. (2014). “Birdsnap: Large-scale fine-grained visual categorization of birds”. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, pp. 2019–2026.
- Branson, Steve et al. (2014). “Improved Bird Species Recognition Using Pose Normalized Deep Convolutional Nets.” In: *BMVC*. Vol. 1. 6, p. 7.
- Chai, Yuning, Victor Lempitsky, and Andrew Zisserman (2013). “Symbiotic segmentation and part localization for fine-grained categorization”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 321–328.
- Deng, Jia et al. (2009). “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pp. 248–255.
- Donahue, Jeff et al. (2014). “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition.” In: *Icml*. Vol. 32, pp. 647–655.
- Elkan, Charles (2001). “The foundations of cost-sensitive learning”. In: *International joint conference on artificial intelligence*. Vol. 17. 1. LAWRENCE ERLBAUM ASSOCIATES LTD, pp. 973–978.
- Everingham, M. and et al. (2005). “The 2005 PASCAL Visual Object Classes Challenge”. In: *First PASCAL Machine Learning Challenges Workshop, MLCW*, pp. 117–176.
- Farrell, Ryan et al. (2011). “Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, pp. 161–168.
- Fei-Fei, Li, R. Fergus, and Pietro Perona (2004). “Learning Generative Visual Models From Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories”. In: *IEEE CVPR Workshop of Generative Model Based Vision (WGBMV)*.
- Fukushima, Kunihiro and Sei Miyake (1982). “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition”. In: *Competition and cooperation in neural nets*. Springer, pp. 267–285.
- Gavves, Efstratios et al. (2015). “Local alignments for fine-grained categorization”. In: *International Journal of Computer Vision* 111.2, pp. 191–212.

- Girshick, Ross et al. (2014). “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.
- Görling, Christoph et al. (2014). “Nonparametric part transfer for fine-grained recognition”. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, pp. 2489–2496.
- Hariharan, Bharath and Ross Girshick. “Low-shot Visual Recognition by Shrinking and Hallucinating Features”. In:
- He, Haibo and Eduardo A Garcia (2009). “Learning from imbalanced data”. In: *Knowledge and Data Engineering, IEEE Transactions on* 21.9, pp. 1263–1284.
- He, Kaiming et al. (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.
- Howard, Andrew G (2013). “Some improvements on deep convolutional neural network based image classification”. In: *arXiv preprint arXiv:1312.5402*.
- Huh, Minyoung, Pulkit Agrawal, and Alexei A Efros (2016). “What makes ImageNet good for transfer learning?” In: *arXiv preprint arXiv:1608.08614*.
- Khosla, Aditya et al. (2011). “Novel Dataset for Fine-Grained Image Categorization”. In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO.
- Krause, Jonathan, Hailin Jin, et al. (2015). “Fine-grained recognition without part annotations”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5546–5555.
- Krause, Jonathan, Benjamin Sapp, et al. (2016). “The unreasonable effectiveness of noisy data for fine-grained recognition”. In: *European Conference on Computer Vision*. Springer, pp. 301–320.
- Krause, Jonathan, Michael Stark, et al. (2013). “3d object representations for fine-grained categorization”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks.” In: *NIPS*.
- Kumar, Neeraj et al. (2012). “Leafsnap: A computer vision system for automatic plant species identification”. In: *Computer Vision—ECCV 2012*. Springer, pp. 502–516.
- LeCun, Yann et al. (1989). “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4, pp. 541–551.
- Lin, Tsung-Yi et al. (2014). “Microsoft COCO: Common objects in context”. In: *ECCV*.

- Lin, Tsung-Yu, Aruni RoyChowdhury, and Subhransu Maji (2015). “Bilinear CNN models for fine-grained visual recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1449–1457.
- Lin, Yen-Liang et al. (2014). “Jointly optimizing 3d model fitting and fine-grained classification”. In: *Computer Vision–ECCV 2014*. Springer, pp. 466–480.
- Lindholm, Erik et al. (2008). “NVIDIA Tesla: A unified graphics and computing architecture”. In: *IEEE micro* 28.2.
- Liu, Jiongxin et al. (2012). “Dog breed classification using part localization”. In: *Computer Vision–ECCV 2012*. Springer, pp. 172–185.
- Long, Mingsheng et al. (2015). “Learning Transferable Features with Deep Adaptation Networks.” In: *ICML*, pp. 97–105.
- Maji, Subhransu et al. (2013). “Fine-grained visual classification of aircraft”. In: *arXiv preprint arXiv:1306.5151*.
- Martin Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: <http://tensorflow.org/>.
- Nilsback, Maria-Elena and Andrew Zisserman (2006). “A visual vocabulary for flower classification”. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. IEEE, pp. 1447–1454.
- Oquab, Maxime et al. (2014). “Learning and transferring mid-level image representations using convolutional neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724.
- Pan, Sinno Jialin and Qiang Yang (2010). “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359.
- Pu, Jian et al. (2014). “Which looks like which: Exploring inter-class relationships in fine-grained visual categorization”. In: *Computer Vision–ECCV 2014*. Springer, pp. 425–440.
- Rippel, Oren et al. (2015). “Metric learning with adaptive density discrimination”. In: *ICLR*.
- Russakovsky, Olga et al. (2015). “Imagenet large scale visual recognition challenge”. In: *International Journal of Computer Vision* 115.3, pp. 211–252.
- Salakhutdinov, Ruslan, Antonio Torralba, and Josh Tenenbaum (2011). “Learning to share visual appearance for multiclass object detection”. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, pp. 1481–1488.
- Sharif Razavian, Ali et al. (2014). “CNN features off-the-shelf: an astounding baseline for recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 806–813.

- Shih, Kevin J et al. (2015). “Part Localization using Multi-Proposal Consensus for Fine-Grained Categorization”. In: *BMVC*.
- Simon, Marcel and Erik Rodner (2015). “Neural activation constellations: Unsupervised part model discovery with convolutional networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1143–1151.
- Sullivan, Brian L et al. (2009). “eBird: A citizen-based bird observation network in the biological sciences”. In: *Biological Conservation* 142.10, pp. 2282–2292.
- Szegedy, Christian et al. (2016). “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.
- Tzeng, Eric et al. (2015). “Simultaneous deep transfer across domains and tasks”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4068–4076.
- Van Horn, Grant et al. (2015). “Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604. DOI: 10.1109/CVPR.2015.7298658.
- Vedaldi, Andrea et al. (2014). “Understanding objects in detail with fine-grained attributes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3622–3629.
- Wah, Catherine et al. (2011). “The caltech-ucsd birds-200-2011 dataset”. In:
- Wang, Yu-Xiong and Martial Hebert (2016a). “Learning from Small Sample Sets by Combining Unsupervised Meta-Training with CNNs”. In: *Advances in Neural Information Processing Systems*, pp. 244–252.
- (2016b). “Learning to learn: Model regression networks for easy small sample learning”. In: *European Conference on Computer Vision*. Springer, pp. 616–634.
- Wegner, Jan D et al. (2016). “Cataloging public objects using aerial and street-level images-urban trees”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6014–6023.
- Welinder, Peter et al. (2010). “Caltech-UCSD birds 200”. In:
- Xiao, Tianjun et al. (2015). “The application of two-level attention models in deep convolutional neural network for fine-grained image classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 842–850.
- Xie, Saining et al. (2015). “Hyper-class augmented and regularized deep learning for fine-grained image classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2645–2654.

- Xu, Zhe et al. (2015). “Augmenting strong supervision using web data for fine-grained categorization”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2524–2532.
- Yosinski, Jason et al. (2014). “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems*, pp. 3320–3328.
- Zhang, Ning et al. (2014). “Part-based R-CNNs for fine-grained category detection”. In: *Computer Vision—ECCV 2014*. Springer, pp. 834–849.
- Zhang, Yu et al. (2016). “Weakly supervised fine-grained categorization with part-based image representation”. In: *IEEE Transactions on Image Processing* 25.4, pp. 1713–1725.
- Zhou, Bolei et al. (2014). “Learning deep features for scene recognition using places database”. In: *Advances in neural information processing systems*, pp. 487–495.
- Zhu, Xiangxin, Dragomir Anguelov, and Deva Ramanan (2014). “Capturing long-tail distributions of object subcategories”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 915–922.

*Chapter 3***LEAN CROWDSOURCING: COMBINING HUMANS AND MACHINES IN AN ONLINE SYSTEM**

Branson, Steve, Grant Van Horn, and Pietro Perona (2017). “Lean Crowdsourcing: Combining Humans and Machines in an Online System”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7474–7483. DOI: 10.1109/CVPR.2017.647.

**3.1 Abstract**

We introduce a method to greatly reduce the amount of redundant annotations required when crowdsourcing annotations, such as bounding boxes, parts, and class labels. For example, if two Mechanical Turkers happen to click on the same pixel location when annotating a part in a given image—an event that is very unlikely to occur by random chance— it is a strong indication that the location is correct. A similar type of confidence can be obtained if a single Turker happened to agree with a computer vision estimate. We thus incrementally collect a variable number of worker annotations per image based on online estimates of confidence. This is done using a sequential estimation of risk over a probabilistic model that combines worker skill, image difficulty, and an incrementally trained computer vision model. We develop specialized models and algorithms for binary annotation, part keypoint annotation, and sets of bounding box annotations. We show that our method can reduce annotation time by a factor of 4-11 for binary filtering of websearch results, 2-4 for annotation of boxes of pedestrians in images, while in many cases also reducing annotation error. We will make an end-to-end version of our system publicly available.

**3.2 Introduction**

Availability of large labeled datasets like ImageNet (Deng, Dong, et al., 2009; Lin et al., 2014) is one of the main catalysts for recent dramatic performance improvement in computer vision (Krizhevsky, Sutskever, and Hinton, 2012; He et al., 2015; Szegedy, W. Liu, et al., 2015; Szegedy, Reed, et al., 2014). While sophisticated crowdsourcing algorithms have been developed for classification (Whitehill et al., 2009; Welinder, Branson, et al., 2010; Welinder and Perona, 2010), there is a



relative lack of methods and publicly available tools that use smarter crowdsourcing algorithms for other types of annotation.

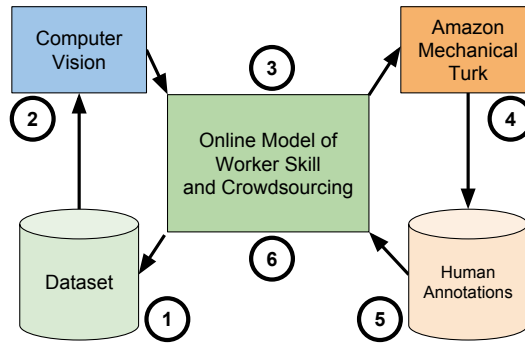


Figure 3.1: A schematic of our proposed method. **(1)** The system is initialized with a dataset of images. Each global step of the method will add annotations to this dataset. **(2)** The computer vision system incrementally retrains using current worker labels. **(3)** The crowdsourcing model updates its predictions of worker skills and image labels and decides which images are finished based on a risk-based quality assurance threshold. Unfinished images are sent to Amazon Mechanical Turk. **(4-5)** Workers on AMT annotate the images. **(6)** The crowdsourcing model continues to update its predictions of worker skills and image labels, and the cycle is repeated until all images are marked as complete.

We have developed a simple-to-use, publicly available tool that incorporates and extends many recent advances in crowdsourcing methods to different types of annotation, like part annotation and multi-object bounding box annotation, and also interfaces directly with Mechanical Turk. One key inspiration is the notion of online crowdsourcing (Welinder and Perona, 2010), where instead of obtaining the same number of annotations for all images, the parameters of the crowdsourcing model are estimated incrementally until a desired confidence level on image labels is achieved. We find that this type of approach is very effective for annotation modalities such as parts and bounding boxes, if one first develops an appropriate probabilistic model of annotation. Second, we develop and test models of worker skill and image difficulty, which we develop for parts, bounding boxes, and binary classification. Further, online crowdsourcing can naturally be extended by machine-in-the-loop methods, where an incrementally-trained, computer vision predictor is another source of information in the online crowdsourcing early stoppage criterion.

Our main contributions are: (1) an online algorithm and stopping criterion for binary, part, and object crowdsourcing, (2) a worker skill and image difficulty crowdsourcing model for binary, part, and object annotations, (3) incorporation of online learning of computer vision algorithms to speedup crowdsourcing, and (4) a

publicly available tool that interfaces with Mechanical Turk and incorporates these algorithms. We show that contributions 1–3 lead to significant improvements in annotation time and/or annotation quality for each type of annotation. For binary classification, annotation error with 1.37 workers per image is lower using our method than when using majority vote and 15 workers per image. For bounding boxes, our method produces lower error with 1.97 workers per image, compared to majority vote using 7 workers per image. For parts, a variation of our system without computer vision was used to annotate accurately a dataset of 11 semantic parts on 55,000 images, averaging 2.3 workers per part.

We note that while incorporating computer vision in the loop speeds up annotation time, computer vision researchers wishing to collect datasets for benchmarking algorithms may choose to toggle off this option to avoid potential issues with bias. At the same time, we believe that it is a very valuable feature in applied settings. For example, a biologist may need to annotate the location of all cells in a dataset of images, not caring if the annotations come from humans or machines, but needing to ensure a certain level of annotation quality. Our method offers an end-to-end tool for collecting training data, training a prediction algorithm, combining human and machine predictions and vetting their quality, while attempting to minimize human time. This may be a useful tool for several applications.

### 3.3 Related Work

Kovashka et al. (A. Kovashka et al., 2016) provide a thorough overview of crowdsourcing in computer vision. Sorokin and Forsyth (Sorokin and Forsyth, 2008) proposed three methods for collecting quality annotations on crowdsourcing platforms. The first is to build a gold standard set (Larlus et al., 2014) to verify work and filter out underperforming workers. The second is to use a grading scheme to evaluate the performance of the workers. This scheme can be accomplished by workers grading each other through a variety of interfaces (Su, Deng, and Fei-Fei, 2012; Russakovsky, L.-J. Li, and Fei-Fei, 2015; Lin et al., 2014), workers grading themselves through monetary incentives (Nihar Bhadresh Shah and Denny Zhou, 2015), and heuristic grading (Russell et al., 2008; Vittayakorn and J. Hays, 2011). The third method resorts to redundantly annotating images and aggregate the results. This has become the standard method for crowdsourcing.

There is a large body of work that explores aggregating worker answers to maximize the accuracy of the estimated labels. Approaches that propose methods to combine

multiple annotations with an assurance on quality are the most similar to our method. Existing work has predominantly used the Dawid-Skene model (Dawid and Skene, 1979). The Dawid-Skene model iteratively infers the reliability of each worker and updates the belief on the true labels. In this setup, individual tasks are assumed to be equally difficult, and researchers often reduce the task to binary classification. Inference algorithms for this model include (Dawid and Skene, 1979; Smyth et al., 1995; Jin and Ghahramani, 2002; Sheng, Provost, and Ipeirotis, 2008; Ghosh, Kale, and McAfee, 2011; Karger, Oh, and D. Shah, 2011; Q. Liu, Peng, and Ihler, 2012; Denny Zhou et al., 2012; H. Li and Yu, 2014; Y. Zhang et al., 2014; Dalvi et al., 2013; Karger, Oh, and D. Shah, 2013; Ok et al., 2016). Some work has focused on theoretical bounds for this setup (Karger, Oh, and D. Shah, 2011; Ghosh, Kale, and McAfee, 2011; Gao and Dengyong Zhou, 2013; Y. Zhang et al., 2014; Dalvi et al., 2013). (Sheng, Provost, and Ipeirotis, 2008; Tian and Zhu, 2015) reconcile multiple annotators through majority voting and worker quality estimates. (Welinder and Perona, 2010; Welinder, Branson, et al., 2010; Long, Hua, and Kapoor, 2013; Wang, Ipeirotis, and Provost, 2013) jointly model labels and the competence of the annotators. (Hua et al., 2013; Long, Hua, and Kapoor, 2013; Long and Hua, 2015) explore the active learning regime of selecting the next data to annotate, as well as which annotator to query. Our approach differs from these previous methods by merging the online notion of (Welinder and Perona, 2010) with the worker modeling of (Welinder, Branson, et al., 2010), and we incorporate a computer vision component as well as provide the framework for performing binary classification, bounding box and part annotations.

In online marketplaces, it is typically unrealistic to assume that all workers are equally adept at a task. The previously listed work specifically tries to model this. However, it is also unrealistic to assume that task difficulty is constant for all tasks. Methods have been developed to model both worker skills and task difficulty (Carpenter, 2008; Raykar et al., 2010; Whitehill et al., 2009; Welinder, Branson, et al., 2010; Snow et al., 2008; Sheng, Provost, and Ipeirotis, 2008; Denny Zhou et al., 2012; Dengyong Zhou et al., 2015), and there is a growing body of work that moves past the Dawid-Skene model (Carpenter, 2008; Whitehill et al., 2009; Welinder, Branson, et al., 2010; Long, Hua, and Kapoor, 2013; Wang, Ipeirotis, and Provost, 2013; Dengyong Zhou et al., 2015; Nihar B Shah, Balakrishnan, and Wainwright, 2016).

Our work is related to human-in-the-loop active learning. Prior work in this area

has contributed methods for tasks such as fine-grained image classification (Branson et al., 2010; Wah, Branson, Perona, et al., 2011; Deng, Krause, and Fei-Fei, 2013; Wah, Van Horn, et al., 2014), image segmentation (Rubinstein, C. Liu, and Freeman, 2012; Dutt Jain and Grauman, 2013; Gurari et al., 2015; Jain and Grauman, 2016), attribute-based classification (A. Kovashka, Vijayanarasimhan, and Grauman, 2011; Parkash and Parikh, 2012; Biswas and Parikh, 2013), image clustering (Lad and Parikh, 2014), image annotation (Vijayanarasimhan and Grauman, 2009a; Vijayanarasimhan and Grauman, 2009b; Siddiquie and Gupta, 2010; Yao et al., 2012; Russakovsky, L.-J. Li, and Fei-Fei, 2015), human interaction (Khodabandeh et al., 2015) and object annotation (Vondrick, D. Patterson, and Ramanan, 2013) and segmentation (Shankar Nagaraja, Schmidt, and Brox, 2015) in videos. For simplicity, we do not incorporate an active learning component when selecting the next batch of images to annotate or question to ask, but this can be included in our framework.

Additional methods to reduce annotation effort include better interfaces, better task organization (Chilton et al., 2013; Deng, Russakovsky, et al., 2014; Wilber, Kwak, and S. J. Belongie, 2014), and gamification (Von Ahn and Dabbish, 2004; Von Ahn and Dabbish, 2005; Kazemzadeh et al., 2014; Deng, Krause, and Fei-Fei, 2013).

Our work is different from the previous work because we combine a worker skill model, a task difficulty model, a computer vision component, and presenting frameworks for binary classification, multi-instance bounding box annotation, and part keypoint annotation. Additional methods have focused on filtering out bad workers (Long, Hua, and Kapoor, 2013; Hua et al., 2013; Long and Hua, 2015) or combining known weak and strong annotators (Chicheng Zhang and Chaudhuri, 2015; Gurari et al., 2015; G. Patterson, G. V. H. S. Belongie, and P. P. J. Hays, 2015), or optimizing the payment to the workers (Wang, Ipeirotis, and Provost, 2013).

### 3.4 Method

Let  $X = \{x_i\}_{i=1}^N$  be a set of images we want to label with unknown true labels  $Y = \{y_i\}_{i=1}^N$  using a pool of imperfect crowd workers. We first describe the problem generally, where depending on the desired application, each  $y_i$  may represent a class label, bounding box, part location, or some other type of semantic annotation. For each image  $i$ , our goal is to recover a label  $\bar{y}_i$  that is equivalent to  $y_i$  with high probability by combining multiple redundant annotations  $Z_i = \{z_{ij}\}_{j=1}^{|\mathcal{W}_i|}$ , where each  $z_{ij}$  is an imperfect worker label (i.e., their perception of  $y_i$ ), and  $\mathcal{W}_i$  is that set of workers that annotated image  $i$ .

Importantly, the number of annotations  $|\mathcal{W}_i|$  can vary significantly for different images  $i$ . This occurs because our confidence in an estimated label  $\bar{y}_i$  will depend not only on the number of redundant annotations  $|\mathcal{W}_i|$ , but also on the level of agreement between those annotations  $Z_i$ , the skill level of the particular workers that annotated  $i$ , and the agreement with a computer vision algorithm (that is incrementally trained). For example, if two different annotators both happened to provide nearly identical bounding box annotations for a given image—the probability of which is very small by random chance—we could be fairly confident of their correctness. On the other hand, we couldn't, as a general rule, collect only two bounding boxes per image, because workers will occasionally make mistakes. Our objective is then to implement an online policy for choosing whether or not to augment each image with additional annotations and to incrementally train a computer vision algorithm with the annotations that have been collected so far.

### Online Crowdsourcing

We first describe a simplified model that does not include a worker skill model or computer vision in the loop. We will augment this simplified model in subsequent sections. At any given time step, let  $Z = \{Z_i\}_{i=1}^N$  be the set of worker annotations for all images. We define the probability over observed images, true labels, and worker labels as  $p(Y, Z) = \prod_i p(y_i) \left( \prod_{j \in \mathcal{W}_i} p(z_{ij}|y_i) \right)$ , where  $p(y_i)$  is a prior probability over possible labels, and  $p(z_{ij}|y_i)$  is a model of noisy worker annotations. Here we have assumed that each worker label is independent. The maximum likelihood solution  $\bar{Y} = \arg \max p(Y|Z) = \arg \max p(Y, Z)$  can be found for each image separately:

$$\bar{y}_i = \arg \max_{y_i} \left( p(y_i) \prod_{j \in \mathcal{W}_i} p(z_{ij}|y_i) \right) \quad (3.1)$$

The risk  $\mathcal{R}(\bar{y}_i) = \int_{y_i} \ell(y_i, \bar{y}_i) p(y_i|Z_i)$  associated with the predicted label is

$$\mathcal{R}(\bar{y}_i) = \frac{\int_{y_i} \ell(y_i, \bar{y}_i) p(y_i) \prod_{j \in \mathcal{W}_i} p(z_{ij}|y_i)}{\int_{y_i} p(y_i) \prod_{j \in \mathcal{W}_i} p(z_{ij}|y_i)} \quad (3.2)$$

where  $\ell(y_i, \bar{y}_i)$  is the loss associated with the predicted label  $\bar{y}_i$  when the true label is  $y_i$ . A logical criterion is to accept  $\bar{y}_i$  once the risk drops below a certain threshold  $\mathcal{R}(\bar{y}_i) \leq \tau_\epsilon$  (i.e.,  $\tau_\epsilon$  is the minimum tolerable error per image). The basic online crowdsourcing algorithm, shown in Algorithm 1, processes images in batches (because sending images to services like Mechanical Turk is easier in batches).

Currently, we give priority to annotating unfinished images with the fewest number of worker annotation  $|\mathcal{W}_i|$ ; however, one could incorporate more sophisticated active learning criteria in future work. Each time a new batch is received, combined image labels  $\bar{y}_i$  are re-estimated, and the risk criterion is used to determine whether or not an image is finished or may require more worker annotations.

---

**Algorithm 1** Online Crowdsourcing
 

---

```

1: input: unlabeled images  $X = \{x_i\}_{i=1}^N$ 
2: Initialize unfinished/finished sets:  $U \leftarrow \{i\}_{i=1}^N$ ,  $F \leftarrow \emptyset$ 
3: Initialize  $\bar{W}$ ,  $\bar{I}$  using prior probabilities
4: repeat
5:   Select a batch  $B \subseteq U$  of unfinished examples
6:   For  $i \in B$  obtain new crowd label  $z_{ij}$ :  $Z_i \leftarrow Z_i \cup z_{ij}$ 
7:   repeat ▷ Max likelihood estimation
8:     Estimate dataset-wide priors  $p(d_i)$ ,  $p(w_j)$ 
9:     Predict true labels:
10:       $\forall_i, \bar{y}_i \leftarrow \arg \max_{y_i} p(y_i|x_i, \bar{\theta})p(Z_i|y_i, \bar{d}_i, \bar{W})$ 
11:      Predict image difficulties:
12:       $\forall_i, \bar{d}_i \leftarrow \arg \max_{d_i} p(d_i)p(Z_i|\bar{y}_i, d_i, \bar{W})$ 
13:      Predict worker parameters:
14:       $\forall_j, \bar{w}_j \leftarrow \arg \max_{w_j} p(w_j) \prod_{i \in \mathcal{I}_j} p(z_{ij}|\bar{y}_i, \bar{d}_i, w_j)$ 
15:   until Until convergence
16:   Using  $K$ -fold cross-validation, train computer vision on dataset  $\{(x_i, \bar{y}_i)\}_{i, |\mathcal{W}_i| > 0}$ ,
    and calibrate probabilities  $p(y_i|x_i, \bar{\theta}_k)$ 
17:   Predict true labels:
18:    $\forall_i, \bar{y}_i \leftarrow \arg \max_{y_i} p(y_i|x_i, \bar{\theta})p(Z_i|y_i, \bar{d}_i, \bar{W})$ 
19:   for  $i \in B$  do ▷ Check for finished labels
20:      $\mathcal{R}_i \leftarrow \frac{\int_{y_i} \ell(y_i, \bar{y}_i) p(\bar{y}_i|x_i, \bar{\theta}) \prod_{j \in \mathcal{W}_i} p(z_{ij}|y_i, d_i, w_j)}{\int_{y_i} p(\bar{y}_i|x_i, \bar{\theta}) \prod_{j \in \mathcal{W}_i} p(z_{ij}|y_i, d_i, w_j)}$ 
21:     if  $\mathcal{R}_i \leq \tau_\epsilon$ :  $F \leftarrow F \cup i$ ,  $U \leftarrow U \setminus i$ 
22:   end for
23: until  $U = \emptyset$ 
24: return  $Y \leftarrow \{\bar{y}_i\}_{i=1}^N$ 

```

---

### Adding Computer Vision

A smarter algorithm can be obtained by using the actual pixel contents  $x_i$  of each image as an additional source of information. We consider two possible approaches: (1) a naive algorithm that treats computer vision the same way as a human worker by appending the computer vision prediction  $z_{i,cv}$  to the set of worker labels  $\mathcal{W}_i$ ,

and (2) a smarter algorithm that exploits the fact that computer vision can provide additional information than a single label output (e.g., confidence estimates that a bounding box occurs at each pixel location in an image).

For the smarter approach, the joint probability over observed images, true labels, and worker labels is:

$$p(Y, Z, \theta | X) = p(\theta) \prod_i \left( p(y_i | x_i, \theta) \prod_{j \in \mathcal{W}_i} p(z_{ij} | y_i) \right) \quad (3.3)$$

where  $p(y_i | x_i, \theta)$  is the estimate of a computer vision algorithm with parameters  $\theta$ . If  $\theta$  is fixed, the predicted label  $\bar{y}_i$  for each image and its associated risk  $\mathcal{R}(\bar{y}_i)$  can be simply found by using the computer vision prediction  $p(y_i | x_i, \theta)$  instead of the prior  $p(y_i)$  in Equations 3.1, 3.2.

### Training Computer Vision:

The main challenge is then training the computer vision system (estimating computer vision parameters  $\theta$ ), given that we incrementally obtain new worker labels over time. While many possible approaches could be used, in practice we retrain the computer vision algorithm each time we obtain a new batch of labels from Mechanical Turk. For each step, we treat the currently predicted labels  $\bar{y}_i$  for each image with at least one worker label  $|\mathcal{W}_i| \geq 1$  as training labels to an off-the-shelf computer vision algorithm. While the predicted labels  $\bar{y}_i$  are clearly very noisy when the number of workers per image is still small, we rely on a post-training probability calibration step to cope with resulting noisy computer vision predictions. We use a modified version of  $K$ -fold cross validation: for each split  $k$ , we use  $(K - 1)/K$  examples for training and the remaining  $(k - 1)/K$  examples for probability calibration. We filter out images with  $|\mathcal{W}_i| < 1$  from both training and probability calibration; however, all  $1/K$  images are used for outputting probability estimates  $p(y_i | x_i, \theta_k)$ , including images with  $|\mathcal{W}_i| = 0$ . This procedure ensures that estimates  $p(y_i | x_i, \theta_k)$  are produced using a model that wasn't trained on labels from image  $i$ .

### Worker Skill and Image Difficulty Model

More sophisticated methods can model the fact that some workers are more skillful or careful than others, and some images are more difficult or ambiguous than others. Let  $W = \{w_j\}_{j=1}^M$  be parameters encoding the skill level of our pool of  $M$  crowd workers, and let  $D = \{d_i\}_{i=1}^n$  be parameters encoding the level of inherent difficulty of annotating each image  $i$  (to this point, we are just defining  $W$  and  $D$  abstractly).

Then the joint probability is

$$p(Y, Z, W, D, \theta | X) = p(\theta) \prod_i (p(d_i) p(y_i | x_i, \theta)) \prod_j p(w_j) \prod_{i,j \in \mathcal{W}_i} p(z_{ij} | y_i, d_i, w_j) \quad (3.4)$$

where  $p(d_i)$  is a prior on the image difficulty,  $p(w_j)$  is a prior on a worker's skill level, and  $p(z_{ij} | y_i, d_i, w_j)$  models noisy worker responses as a function of the ground truth label, image difficulty, and worker skill parameters. Let  $\bar{Y}, \bar{W}, \bar{D}, \bar{\theta} = \arg \max_{Y, W, D, \theta} p(Y, W, D, \theta | X, Z)$  be the maximum likelihood solution to Eq. 3.4: In practice, we estimate parameters using alternating maximization algorithms, where we optimize with respect to the parameters of one image or worker at a time (often with fast analytical solutions):

$$\bar{y}_i = \arg \max_{y_i} p(y_i | x_i, \bar{\theta}) \prod_{j \in \mathcal{W}_i} p(z_{ij} | y_i, d_i, w_j) \quad (3.5)$$

$$\bar{d}_i = \arg \max_{d_i} p(d_i) \prod_{j \in \mathcal{W}_i} p(z_{ij} | y_i, d_i, w_j) \quad (3.6)$$

$$\bar{w}_j = \arg \max_{w_j} p(w_j) \prod_{i \in \mathcal{I}_j} p(z_{ij} | \bar{y}_i, \bar{d}_i, w_j) \quad (3.7)$$

$$\bar{\theta} = \arg \max_{\theta} p(\theta) \prod_i p(\bar{y}_i | x_i, \theta) \quad (3.8)$$

where  $\mathcal{I}_j$  is the set of images labeled by worker  $j$ . Exact computation of the risk  $\mathcal{R}_i = \mathcal{R}(\bar{y}_i)$  is difficult because labels for different images are correlated through  $W$  and  $\theta$ . An estimate is to assume our approximations  $\bar{W}$ ,  $\bar{I}$ , and  $\bar{\theta}$  are good enough  $\mathcal{R}(\bar{y}_i) \approx \int_{y_i} \ell(y_i, \bar{y}_i) p(y_i | X, Z, \bar{\theta}, \bar{W}, \bar{D})$

$$\mathcal{R}(\bar{y}_i) \approx \frac{\int_{y_i} \ell(y_i, \bar{y}_i) p(y_i | x_i, \bar{\theta}) \prod_{j \in \mathcal{W}_i} p(z_{ij} | y_i, \bar{d}_i, \bar{w}_j)}{\int_{y_i} p(y_i | x_i, \bar{\theta}) \prod_{j \in \mathcal{W}_i} p(z_{ij} | y_i, \bar{d}_i, \bar{w}_j)}$$

such that Eq. 3.9 can be solved separately for each image  $i$ .

### Considerations in designing priors:

Incorporating priors is important to make the system more robust. Due to the online nature of the algorithm, in early batches the number of images  $|\mathcal{I}_j|$  annotated by each worker  $j$  is likely small, making worker skill  $w_j$  difficult to estimate. Additionally, in practice many images will satisfy the minimum risk criterion with two or less labels  $|\mathcal{W}_i| \leq 2$ , making image difficulty  $d_i$  difficult to estimate. In practice we use a tiered prior system. A dataset-wide worker skill prior  $p(w_j)$  and image difficulty



prior  $p(d_i)$  (treating all workers and images the same) is estimated and used to regularize per worker and per image parameters when the number of annotations is small. As a heuristic to avoid over-estimating skills, we restrict ourselves to considering images with at least 2 worker labels  $|\mathcal{W}_i| > 1$  when learning worker skills, image difficulties, and their priors, since agreement between worker labels is the only viable signal for estimating worker skill. We also employ a hand-coded prior that regularizes the learned dataset-wide priors. We will describe what this means specifically in subsequent sections when we describe each type of annotation.

### 3.5 Models For Common Types of Annotations

Algorithm 1 provides pseudo-code to implement the online crowdsourcing algorithm for any type of annotation. Supporting a new type of annotation involves defining how to represent true labels  $y_i$  and worker annotations  $z_{ij}$ , and implementing solvers for inferring the (1) true labels  $\bar{y}_i$  (Eq. 3.5), (2) image difficulties  $\bar{d}_i$  (Eq. 3.6), (3) worker skills  $\bar{w}_j$  (Eq. 3.7), (4) computer vision parameters  $\bar{\theta}$  (Eq. 3.8), and (5) risk  $\mathcal{R}_i$  associated with the predicted true label (Eq. 3.9). Although this is somewhat involved, we note that each component 2-5 individually (though useful in practice) could optionally be omitted. In the sections below, we detail models and algorithms for 3 common types of annotations: class labels, part annotations, and multi-object bounding box labels. These cover several building blocks for different annotation types including boolean variables, continuous variables, and unordered sets.

### 3.6 Binary Annotation

Here, each label  $y_i \in \{0, 1\}$ , denotes the absence/presence of a class of interest. This is the simplest type of annotation and has also been covered most extensively in crowdsourcing literature, therefore we try to keep this section as simple as possible. At the same time, many important datasets such as ImageNet (Deng, Dong, et al., 2009) and Caltech-256 (Griffin, Holub, and Perona, 2007) are obtained by binary filtering of image search results, and we are unaware of an existing crowdsourcing tool that incorporates online crowdsourcing with a worker skill model (not to mention computer vision), so we believe our binary annotation tool is worthwhile.

#### Binary worker skill model:

We model worker skill  $w_j = [p_j^1, p_j^0]$  using two parameters representing the worker's skill at identifying true positives and true negatives, respectively. Here, we assume  $z_{ij}$  given  $y_i$  is Bernoulli, such that  $p(z_{ij}|y_i = 1) = p_j^1$  and  $p(z_{ij}|y_i = 0) = p_j^0$ . As

described in Section 3.4, we use a tiered set of priors to make the system robust in corner case settings where there are few workers or images. Ignoring worker identity and assuming a worker label  $z$  given  $y$  is Bernoulli such that  $p(z|y = 1) = p^1$  and  $p(z|y = 0) = p^0$ , we add Beta priors  $\text{Beta}(n_\beta p^0, n_\beta(1 - p^0))$  and  $\text{Beta}(n_\beta p^1, n_\beta(1 - p^1))$  on  $p_j^0$  and  $p_j^1$ , respectively, where  $n_\beta$  is the strength of the prior. An intuition of this is that worker  $j$ 's own labels  $z_{ij}$  softly start to dominate estimation of  $w_j$  once she has labeled more than  $n_\beta$  images, otherwise the dataset-wide priors dominate. We also place Beta priors  $\text{Beta}(n_\beta p, n_\beta(1 - p))$  on  $p^0$  and  $p^1$  to handle cases such as the first couple batches of Algorithm 1. In our implementation, we use  $p = .8$  as a general fairly conservative prior on binary variables and  $n_\beta = 5$ . This model results in simple estimation of worker skill priors  $p(w_j)$  in line 8 of Algorithm 1 by counting the number of labels agreeing with combined predictions:

$$p^k = \frac{n_\beta p + \sum_{ij} 1[z_{ij} = \bar{y}_i = k, |\mathcal{W}_i| > 1]}{n_\beta + \sum_{ij} 1[\bar{y}_i = k, |\mathcal{W}_i| > 1]}, \quad k = 0, 1 \quad (3.9)$$

where  $1[\cdot]$  is the indicator function. Analogously, we estimate worker skills  $w_j$  in line 11 of Algorithm 1 by counting worker  $j$ 's labels that agree with combined predictions:

$$p_j^k = \frac{n_\beta p^k + \sum_{i \in \mathcal{I}_j} 1[z_{ij} = \bar{y}_i = k, |\mathcal{W}_i| > 1]}{n_\beta + \sum_{i \in \mathcal{I}_j} 1[\bar{y}_i = k, |\mathcal{W}_i| > 1]}, \quad k = 0, 1 \quad (3.10)$$

For simplicity, we decided to omit a notion of image difficulty in our binary model after experimentally finding that our simple model was competitive with more sophisticated models like CUBAM (Welinder, Branson, et al., 2010) on most datasets.

### Binary computer vision model:

We use a simple computer vision model based on training a linear SVM on features from a general purpose pre-trained CNN feature extractor (our implementation uses VGG), followed by probability calibration using Platt scaling (Platt et al., 1999) with the validation splits described in Sec. 3.4. This results in probability estimates  $p(y_i|x_i, \theta) = \sigma(\gamma \theta \cdot \phi(x_i))$  for each image  $i$ , where  $\phi(x_i)$  is a CNN feature vector,  $\theta$  is a learned SVM weight vector,  $\gamma$  is probability calibration scalar from Platt scaling, and  $\sigma(\cdot)$  is the sigmoid function. This simple procedure is easily fast enough to run in time less than the time to annotate a batch on Mechanical Turk and is reasonably general purpose.

### 3.7 Part Keypoint Annotation

Part keypoint annotations are popular in computer vision and included in datasets such as MSCOCO (Lin et al., 2014), MPII human pose (Andriluka et al., 2014), and CUB-200-2011 (Wah, Branson, Welinder, et al., 2011). Here, each part is typically represented as an  $x, y$  pixel location  $l$  and binary visibility variable  $v$ , such that  $y_i = (l_i, v_i)$ . While we can model  $v$  using the exact same model as for binary classification (Section 3.6),  $l$  is a continuous variable that necessitates different models. For simplicity, even though most datasets contain several semantic parts of an object, we model and collect each part independently. This simplifies notation and collection; in our experience, Mechanical Turkers tend to be faster/better at annotating a single part in many images than multiple parts in the same image.

We first note that modeling keypoint visibility as binary classification results in two worker skill parameters  $p_j^0$  and  $p_j^1$ , which correspond to the probability that the worker thinks a part is visible when  $v_i = 0$  and  $v_i = 1$ , respectively—these encode different annotators’ tendencies and biases in annotating a part’s visibility. The reader can refer to Eqs. 3.10 and 3.9 for computation of  $p_j^0, p_j^1$  and their priors.

#### Keypoint worker skill image difficulty model:

Let  $l_i$  be the true location of a keypoint in image  $i$ , while  $l_{ij}$  is the location clicked by worker  $j$ . We assume  $l_{ij}$  is Gaussian distributed around  $l_i$  with variance  $\sigma_{ij}^2$ . This variance is governed by the worker’s skill or image difficulty  $\sigma_{ij}^2 = e_{ij}\sigma_j^2 + (1 - e_{ij})\sigma_i^2$ , where  $\sigma_j^2$  represents worker noise (e.g., some workers are more precise than others) and  $\sigma_i^2$  represents per image noise (e.g., the precise location of a bird’s belly in a given image maybe inherently ambiguous), and  $e_{ij}$  is a binary variable that determines if the variance will be governed by worker skill or image difficulty. However, worker  $j$  sometimes makes a gross mistake and clicks somewhere very far from the Gaussian center (e.g., worker  $j$  could be a spammer or could have accidentally clicked an invalid location).  $m_{ij}$  indicates whether or not  $j$  made a mistake—with probability  $p_j^m$ —in which case  $l_{ij}$  is uniformly distributed in the image. Thus

$$p(l_{ij}|y_i, d_i, w_j) = \sum_{m_{ij} \in \{0,1\}} p(m_{ij}|p_j^m) p(l_{ij}|l_i, m_{ij}, \sigma_{ij}) \quad (3.11)$$

where  $p(m_{ij}|p_j^m) = m_{ij}p_j^m + (1 - m_{ij})(1 - p_j^m)$ ,  $p(l_{ij}|l_i, m_{ij}, \sigma_{ij}) = \frac{e_{ij}}{|x_i|} + (1 - e_{ij})g(\|l_{ij} - l_i\|^2; \sigma_{ij}^2)$ ,  $|x_i|$  is the number of pixel locations in  $i$ , and  $g(x^2; \sigma^2)$  is the probability density function for the normal distribution. In summary, we have 4 worker skill parameters  $w_j = [\sigma_j, p_j^m, p_j^0, p_j^1]$  describing errors in clicking precise locations, the

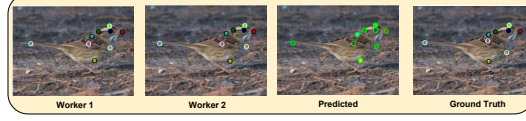


Figure 3.2: Example part annotation sequence showing the common situation where the responses from 2 workers correlate well and are enough for the system to mark the images as finished.

probability of making a mistake, probabilities of correctly identifying visibility, and one image difficulty parameter  $d_i = \sigma_i$  describing ambiguity of the exact keypoint location in the image. As described in Section 3.6, we place a dataset-wide Beta prior  $\text{Beta}(n_\beta p^m, n_\beta(1 - p^m))$  on  $p_j^m$ , where  $p^m$  is a worker agnostic probability of making a mistake and an additional Beta prior  $\text{Beta}(n_\beta p, n_\beta(1 - p))$  on  $p$ . Similarly, we place Scaled inverse chi-squared priors on  $\sigma_j^2$  and  $\sigma_i^2$ , such that  $\sigma_j^2 \sim \text{scale-inv-}\chi^2(n_\beta, \sigma^2)$  and  $\sigma_i^2 \sim \text{scale-inv-}\chi^2(n_\beta, \sigma^2)$  where  $\sigma^2$  is a dataset-wide variance in click location.

### Inferring worker and image parameters:

These priors would lead to simple analytical solutions toward inferring the maximum likelihood image difficulties (Eq. 3.6) and worker skills (Eq. 3.7), if  $m_{ij}$ ,  $e_{ij}$ , and  $\theta$  were known. In practice, we handle latent variables  $m_{ij}$  and  $e_{ij}$  using expectation maximization, with the maximization step over all worker and image parameters, such that worker skill parameters are estimated as

$$\sigma_i^2 = \frac{n_\beta \sigma^2 + \sum_{j \in \mathcal{W}_i} (1 - \mathbb{E}e_{ij})(1 - \mathbb{E}m_{ij}) \|l_{ij} - l_i\|^2}{n_\beta + 2 + \sum_{j \in \mathcal{W}_i} (1 - \mathbb{E}e_{ij})(1 - \mathbb{E}m_{ij})} \quad (3.12)$$

$$\sigma_j^2 = \frac{n_\beta \sigma^2 + \sum_{i \in \mathcal{I}_j} \mathbb{E}e_{ij} (1 - \mathbb{E}m_{ij}) \|l_{ij} - l_i\|^2}{n + 2 + \sum_{i \in \mathcal{I}_j} \mathbb{E}e_{ij} (1 - \mathbb{E}m_{ij})} \quad (3.13)$$

$$p_j^m = \frac{n_\beta p^m + \sum_{i \in \mathcal{I}_j} \mathbb{E}m_{ij}}{n_\beta + |\mathcal{I}_j|} \quad (3.14)$$

These expressions all have intuitive meaning of being like standard empirical estimates of variance or binomial parameters, except that each example might be soft-weighted by  $\mathbb{E}m_{ij}$  or  $\mathbb{E}e_{ij}$ , and  $n_\beta$  synthetic examples have been added from the global prior distribution. Expectations are then

$$\begin{aligned} \mathbb{E}e_{ij} &= \frac{g_j}{g_i + g_j}, & \mathbb{E}m_{ij} &= \frac{1/|x_i|}{1/|x_i| + (1 - \mathbb{E}e_{ij})g_i + \mathbb{E}e_{ij}g_j} \\ g_i &= g(\|l_{ij} - l_i\|^2; \sigma_i^2), & g_j &= g(\|l_{ij} - l_i\|^2; \sigma_j^2) \end{aligned} \quad (3.15)$$

We alternate between maximization and expectation steps, where we initialize with  $\mathbb{E}m_{ij} = 0$  (i.e., assuming an annotator didn't make a mistake) and  $\mathbb{E}e_{ij} = .5$  (i.e., assuming worker noise and image difficulty have equal contribution).

### Inferring true labels:

Inferring  $\bar{y}_i$  (Eq. 3.5) must be done in a more brute-force way due to the presence of the computer vision term  $p(y_i|x_i, \theta)$ . Let  $\mathbb{X}_i$  be a vector of length  $|x_i|$  that stores a probabilistic part detection map; that is, it stores the value of  $p(y_i|x_i, \theta)$  for each possible value of  $y_i$ . Let  $\mathbb{Z}_{ij}$  be a corresponding vector of length  $|x_i|$  that stores the value of  $p(z_{ij}|y_i, d_i, w_j)$  at each pixel location (computed using Eq. 3.11<sup>1</sup>). Then the vector  $\mathbb{Y}_i = \mathbb{X}_i \prod_{j \in \mathcal{W}_i} \mathbb{Z}_{ij}$  densely stores the likelihood of all possible values of  $y_i$ , where products are assumed to be computed using component-wise multiplication. The maximum likelihood label  $\bar{y}_i$  is simply the argmax of  $\mathbb{Y}_i$ .

### Computing risk:

Let  $\mathbb{L}_i$  be a vector of length  $|x_i|$  that stores the loss  $\ell(y_i, \bar{y}_i)$  for each possible value of  $y_i$ . We assume a part prediction is incorrect if its distance from ground truth is bigger than some radius (in practice, we compute the standard deviation of Mechanical Turker click responses on a per part basis and set the radius equal to 2 standard deviations). The risk associated with predicted label  $\bar{y}_i$  according to Eq. 3.9 is then  $\mathcal{R}_i = \mathbb{L}_i^T \mathbb{Y}_i / \|\mathbb{Y}_i\|_1$ .

### Computer Vision:

We can use any detection system that can produce dense detection scores for pixel locations in the image, such as a fully convolutional CNN. The part detection scores can be converted to probabilities using cross-validation, such that part detection scores  $m(x_i, l_i; \theta)$  are converted to probabilities

$$p(y_i|x_i, \theta) = \frac{\exp\{\gamma m(x_i, y_i; \theta)\}}{\sum_{l_i} \exp\{\gamma m(x_i, l_i; \theta)\}} \quad (3.16)$$

with  $\gamma$  learned to maximize the likelihood on the validation set.

## 3.8 Multi-Object Bounding Box Annotations

Similar types of models that were used for part keypoints can be applied to other types of continuous annotations like bounding boxes. However, a significant new challenge is introduced if multiple objects are present in the image, such that each

<sup>1</sup>In practice, we replace  $e_{ij}$  and  $m_{ij}$  with  $\mathbb{E}e_{ij}$  and  $\mathbb{E}m_{ij}$  in Eq. 3.11, which corresponds to marginalizing over latent variables  $e_{ij}$  and  $m_{ij}$ , instead of using maximum likelihood estimates.

worker may label a different number of bounding boxes and may label objects in a different order. Checking for finished labels means ensuring not only that the boundaries of each box are accurate, but also that there are no false negatives or false positives.

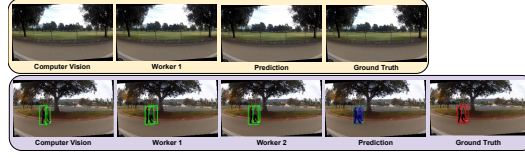


Figure 3.3: Bounding box annotation sequences. The top sequence highlights a good case where only the computer vision system and one human are needed to finish the image. The bottom sequence highlights the average case where two workers and the computer vision system are needed to finish the image.

### Bounding box worker skill and image difficulty model

An image annotation  $y_i = \{b_i^r\}_{r=1}^{|B_i|}$  is composed of a set of objects in the image where box  $b_i^r$  is composed of  $x, y, x2, y2$  coordinates. Worker  $j$ 's corresponding annotation  $z_{ij} = \{b_{ij}^k\}_{k=1}^{|B_{ij}|}$  is composed of a potentially different number  $|B_{ij}|$  of box locations with different ordering. However, if we can predict latent assignments  $\{a_{ij}^k\}_{k=1}^{|B_{ij}|}$ , where  $b_{ij}^k$  is worker  $j$ 's perception of true box  $b_i^{a_{ij}^k}$ , we can model annotation of a matched bounding box exactly as for keypoints, where 2D vectors  $l$  have been replaced by 4D vectors  $b$ .

Thus, as for keypoints the difficulty of image  $i$  is represented by a set of bounding box difficulties:  $d_i = \{\sigma_i^r\}_{r=1}^{|B_i|}$ , which measure to what extent the boundaries of each object in the image are inherently ambiguous. A worker's skill  $w_j = \{p_j^{\text{fp}}, p_j^{\text{fn}}, \sigma_j\}$  encodes the probability  $p_j^{\text{fp}}$  that an annotated box  $b_{ij}^k$  is a false positive (i.e.,  $a_{ij}^k = \emptyset$ ), the probability  $p_j^{\text{fn}}$  that a ground truth box  $b_i^r$  is a false negative (i.e.,  $\forall_k, a_{ij}^k \neq r$ ), and the worker's variance  $\sigma_j^2$  in annotating the exact boundary of a box is modeled as in Section 3.7. The number of true positives  $n_{\text{tp}}$ , false positives  $n_{\text{fp}}$ , and false negatives  $n_{\text{fn}}$  can be written as  $n_{\text{tp}} = \sum_{k=1}^{|B_{ij}|} 1[a_{ij}^k \neq \emptyset]$ ,  $n_{\text{fn}} = |B_i| - n_{\text{tp}}$ ,  $n_{\text{fp}} = |B_{ij}| - n_{\text{tp}}$ . This leads to annotation probabilities

$$p(z_{ij}|y_i, d_i, w_j) = \prod_{k=1 \dots B_{ij}, a_{ij}^k \neq \emptyset} g\left(\left|b_i^{a_{ij}^k} - b_{ij}^k\right|^2; \sigma_{ij}^{k^2}\right) (p_j^{\text{fn}})^{n_{\text{fn}}} (1 - p_j^{\text{fn}})^{n_{\text{tp}}} (p_j^{\text{fp}})^{n_{\text{fp}}} (1 - p_j^{\text{fp}})^{n_{\text{tp}}} \quad (3.17)$$

As in the previous sections, we place dataset-wide priors on all worker and image parameters.

### Computer vision

We train a computer vision detector based on MSC-MultiBox (Szegedy, Reed, et al., 2014), which computes a shortlist of possible object detections and associated detection scores:  $\{(b_{i,cv}^k, m_{i,cv}^k)\}_{k=1}^{|B_{i,cv}|}$ . We choose to treat computer vision like a worker, with learned parameters  $[p_{cv}^{fp}, p_{cv}^{fn}, \sigma_{cv}]$ . The main difference is that we replace the false positive parameter  $p_{cv}^{fp}$  with a per bounding box prediction of the probability of correctness as a function of its detection score  $m_{i,cv}^k$ . The shortlist of detections is first matched to boxes in the predicted label  $\bar{y}_i = \{b_i^r\}_{r=1}^{|B_i|}$ . Let  $r_{i,cv}^k$  be 1 or  $-1$  if detected box  $b_{i,cv}^k$  was matched or unmatched to a box in  $\bar{y}_i$ . Detection scores are converted to probabilities using Platt scaling and the validation sets described in Section 3.4.

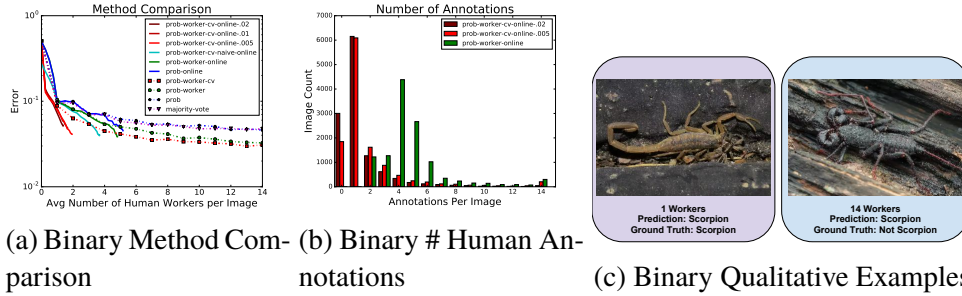


Figure 3.4: **Crowdsourcing Binary Classification Annotations:** (a) Comparison of methods. Our full model prob-worker-cvonline-0.02 obtains results as good as typical baselines with 15 workers (majority-vote and prob) using only 1.37 workers per image on average. (b) Histogram of the number of human annotations required for each image. (c) The image on the left represents an average annotation situation where only the computer vision label and one worker label are needed to confidently label the image. The image on the right (which is not a scorpion) represents a difficult case in which many workers disagreed on the label.

### Inferring true labels and assignments

We devise an approximate algorithm to solve for the maximize likelihood label  $\bar{y}_i$  (Eq. 3.5) concurrently with solving for the best assignment variables  $a_{ij}^k$  between worker and ground truth bounding boxes:

$$\bar{y}_i, a_i = \arg \max_{y_i, a_i} \log \sum_{j \in \mathcal{W}_i} \log p(z_{ij} | y_i, d_i, w_j) \quad (3.18)$$

where  $p(z_{ij}|y_i, d_i, w_j)$  is defined in Eq. 3.17. We formulate the problem as a facility location problem Erlenkotter, 1978, a type of clustering problem where the objective is to choose a set of "facilities" to open up given that each "city" must be connected to a single facility. One can assign custom costs for opening each facility and connecting a given city to a given facility. Simple greedy algorithms are known to have good approximation guarantees for some facility location problems. In our formulation, facilities will be boxes selected to add to the predicted combined label  $\bar{y}_i$ , and city-facility costs will be costs associated with assigning a worker box to an opened box. Due to space limitations we omit derivation details; however, we set facility open costs  $C^{\text{open}}(b_{ij}^k) = \sum_{j \in \mathcal{W}_i} -\log p_j^{\text{fn}}$  and city-facility costs  $C^{\text{match}}(b_{ij}^k, b_{ij'}^{k'}) = -\log(1 - p_j^{\text{fn}}) + \log p_j^{\text{fn}} - \log(1 - p_j^{\text{fp}}) - \log g(\|b_{ij}^k - b_{ij'}^{k'}\|^2; \sigma_j^2)$  for matching worker box  $b_{ij}^k$  to facility  $b_{ij'}^{k'}$ , while not allowing connections where  $j = j'$  unless  $k = k', j = j'$ . We add a dummy facility with open cost 0, such that cities matched to it correspond to worker boxes that are false positives:  $C^{\text{match}}(b_{ij}^k, \text{dummy}) = -\log p_j^{\text{fp}}$ .

### Computing risk

We assume that the loss  $\ell(\bar{y}_i, y_i)$  for annotating bounding boxes is defined as the number of false positive bounding boxes plus the number of false negatives, where boxes match if their area of intersection over union is at least 50%. Previously in this section, we described a procedure for inferring assignments  $\{a_{ij}^k\}_{k=1}^{|B_{ij}|}$  between boxes in each worker label  $z_{ij} = \{b_{ij}^k\}_{k=1}^{|B_{ij}|}$  and predicted combined labels  $\bar{y}_i = \{\bar{b}_i^r\}_{r=1}^{\bar{B}_i}$ . To simplify calculation of risk, we assume our inferred correspondences are valid. In this case, the probability  $p(\text{fp}(\bar{b}_i^r))$  that a box  $\bar{b}_i^r \in \bar{y}_i$  is a false positive can be computed by evaluating the likelihood of worker labels with and without  $\bar{b}_i^r$ :

$$p(\text{fp}(\bar{b}_i^r)) = \frac{\prod_{j \in \mathcal{W}_{ij}} p(z_{ij}|\bar{y}_i \setminus \bar{b}_i^r, d_i, w_j)}{\prod_{j \in \mathcal{W}_{ij}} p(z_{ij}|\bar{y}_i \setminus \bar{b}_i^r, d_i, w_j) + \prod_{j \in \mathcal{W}_{ij}} p(z_{ij}|\bar{y}_i, d_i, w_j)} \quad (3.19)$$

$$= \frac{1}{1 + \prod_{j \in \mathcal{W}_{ij}} \frac{p(z_{ij}|\bar{y}_i, d_i, w_j)}{p(z_{ij}|\bar{y}_i \setminus \bar{b}_i^r, d_i, w_j)}} \quad (3.20)$$

$$= \frac{1}{1 + \prod_{j \in \mathcal{W}_{ij}} p_j^{\text{fn}} \prod_{k, a_{ij}^k=r} \frac{(1-p_j^{\text{fn}})(1-p_j^{\text{fp}})g(b_{ij}^k; \bar{b}_i^r, \sigma_{ij}^{k2})}{p_j^{\text{fn}} p_j^{\text{fp}}}} \quad (3.21)$$

where the second line was found by substituting in  $p(z_{ij}|\bar{y}_i, d_i, w_j)$  as defined in Eq. 18 of the main paper. Computing the expected number of false negatives is more complicated, because it involves considering each possible bounding box location



$\bar{b}_i^{r'}$  in the image (not just in  $\bar{y}_i$ ), and evaluating the relative likelihood of all worker labels if  $\bar{b}_i^{r'}$  were added  $\bar{y}_i$ , so  $p(\text{fn}(\bar{b}_i^{r'})) =$ :

$$\frac{\prod_{j \in \mathcal{W}_{ij}} p(z_{ij} | \bar{y}_i \cup \bar{b}_i^{r'}, d_i, w_j)}{\prod_{j \in \mathcal{W}_{ij}} p(z_{ij} | \bar{y}_i \cup \bar{b}_i^{r'}, d_i, w_j) + \prod_{j \in \mathcal{W}_{ij}} p(z_{ij} | \bar{y}_i, d_i, w_j)} \quad (3.22)$$

$$= 1 - \frac{\prod_{j \in \mathcal{W}_{ij}} p(z_{ij} | \bar{y}_i, d_i, w_j)}{\prod_{j \in \mathcal{W}_{ij}} p(z_{ij} | \bar{y}_i, d_i, w_j) + \prod_{j \in \mathcal{W}_{ij}} p(z_{ij} | \bar{y}_i \cup \bar{b}_i^{r'}, d_i, w_j)} \quad (3.23)$$

$$= 1 - \frac{1}{1 + \prod_{j \in \mathcal{W}_{ij}} p_j^{\text{fn}} \prod_{k, a'_{ij}^k = r'} \frac{(1-p_j^{\text{fn}})(1-p_j^{\text{fp}})g(b_{ij}^k; \bar{b}_i^{r'}, \sigma_{ij}^{k^2})}{p_j^{\text{fn}} p_j^{\text{fp}}}} \quad (3.24)$$

where  $a'_{ij}^k = r'$  represents worker boxes that would be assigned to  $b_i^{r'}$  if it existed in the true label. To infer these assignments, we extend the facility location algorithm. Note that in this algorithm, some worker boxes  $b_{ij}^k$  may be assigned to the dummy facility  $a_{ij}^k = \emptyset$ , which represents worker boxes that are inferred to be false positives. Such worker boxes could also be interpreted as providing probabilistic evidence that a box may occur in a nearby location. We setup a second facility location problem where we enumerate all possible values of  $b_i^{r'}$  (in practice, we incrementally grow a big set of all possible bounding box locations  $B_i^{\text{big}}$ , adding a new one if its intersection over union is at least 50% from all previous boxes in the set). Each  $b_i^{r'}$  in this set is a possible facility with open cost  $C^{\text{open}}(b_i^{r'}) = -\log \prod_{j \in \mathcal{W}_i} p_j^{\text{fn}}$ , and each unassigned worker box  $b_{ij}^k$  with  $a_{ij}^k = \emptyset$  is a city that can be connected to it  $C^{\text{match}}(b_{ij}^k, b_i^{r'}) = -\log \left( \frac{(1-p_j^{\text{fn}})(1-p_j^{\text{fp}})g(b_{ij}^k; b_i^{r'}, \sigma_{ij}^{k^2})}{p_j^{\text{fn}}} \right)$ . This procedure allows us to infer assignments  $a'_{ij}^k = r'$  such that evaluating Eq. 3.24 to estimate  $p(\text{fn}(\bar{b}_i^{r'}))$  is possible. The facility location algorithm can be understood as a way of predicting correspondences/assignments between worker boxes (since each worker may label objects in a different order); the criterion to infer these correspondences is to minimize the negative log-likelihood of all worker labels.

The last consideration is that predicted boxes where the boundaries are too inaccurate will incur both a false positive and false negative according to our loss function. Again assuming that assignments  $a_{ij}^k$  between worker boxes and predicted boxes are correct, the probability  $p(\text{bnd\_off}(\bar{b}_i^{r'}))$  that a predicted box  $\bar{b}_i^{r'}$  has boundaries that

are too far off is:

$$p(\text{bnd\_off}(\bar{b}_i^r)) = \int_{b, \text{IOU}(b, \bar{b}_i^r) > .5} \prod_{jk, j \in \mathcal{W}_i, a_{ij}^k = r} g(b_{ij}^k; b, \sigma_{ij}^2) db \quad (3.25)$$

$$\approx \int_{b, \frac{\|b - \bar{b}_i^r\|^2}{\|\bar{b}_i^r\|^2} > .5} \prod_{jk, j \in \mathcal{W}_i, a_{ij}^k = r} g(b_{ij}^k; b, \sigma_{ij}^2) db \quad (3.26)$$

$$= 1 - \text{erf} \left( .5 \sqrt{\sum_{jk, j \in \mathcal{W}_i, a_{ij}^k = r} \frac{\|\bar{b}_i^r\|^2}{\sigma_{ij}^2}} \right) \quad (3.27)$$

where  $g(x; \mu, \sigma^2)$  is the density function of the Normal distribution and  $\text{erf}(x)$  is the error function. In the 2nd line, we use an approximation that the region where the intersection over union of two boxes is greater than a threshold is similar to the region where their Euclidean distance is greater than a threshold. This enables the integral to have a simple analytical solution.

The total risk  $\mathcal{R}_i$  is then the expected number of false positives (computed by summing over each  $b_i^r$  and computing  $p(\text{fp}(\bar{b}_i^r))$  according to Eq. 3.21), the expected number of false negatives (computed by summing  $p(\text{fn}(\bar{b}_i^r))$  over all possible boxes  $b_i^{r'}$  in the image according to Eq. 3.24), and the expected number of true positives that were too inaccurate to meet the area of intersection over union criterion (computed by summing over each  $b_i^r$  and computing  $p(\text{bnd\_off}(\bar{b}_i^r))$  using Eq. 3.27):

$$\mathcal{R}_i = \sum_{r=1}^{|\bar{B}_i|} p(\text{fp}(\bar{b}_i^r)) + \sum_{r'=1}^{|\bar{B}_i^{\text{big}}|} p(\text{fn}(\bar{b}_i^{r'})) + \sum_{r=1}^{|\bar{B}_i|} (1 - p(\text{fp}(\bar{b}_i^r))) p(\text{bnd\_off}(\bar{b}_i^r)) \quad (3.28)$$

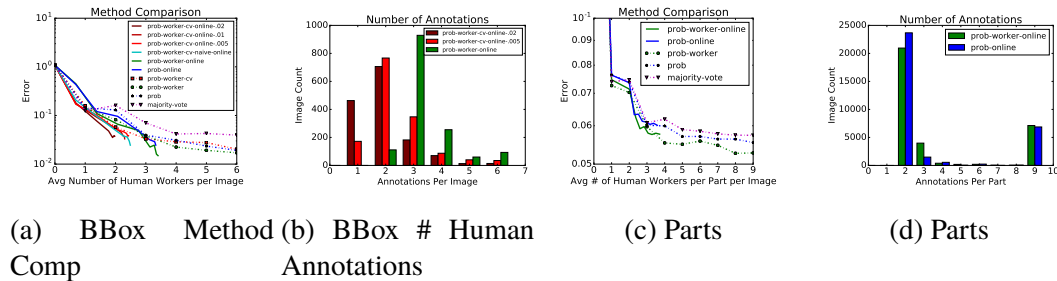
### 3.9 Experiments

We used a live version of our method to collect parts for the NABirds dataset (see Chapter 5). Additionally, we performed ablation studies on datasets for binary, part, and bounding box annotation based on simulating results from real-life MTurk worker annotations.

#### Evaluation Protocol

For each image, we collected an over-abundance of MTurk annotations per image, which were used to simulate results by adding MTurk annotations in random order. The online crowdsourcing algorithm chose whether or not to terminate receiving additional annotations. For lesion studies, we crippled portions of Algorithm 1

as follows: (1) we removed online crowdsourcing by simply running lines 7-14 over the whole dataset with  $k$  workers per image and sweeping over choices of  $k$ , (2) we removed the worker skill, image difficulty model by using dataset-wide priors, and (3) we removed computer vision by using label priors  $p(y_i)$  instead of computer vision estimates  $p(y_i|x_i, \theta)$ . As a baseline, the *majority-vote* method in plots 3.4a,3.5a,3.5c shows what we consider to be the most standard and commonly used method/baseline for crowdsourcing. For binary annotation, this selects the label with the most worker votes. For parts, it selects the median worker part location (i.e., the one that matches the most other worker annotations with minimal loss). The same basic method is used for bounding boxes, adding a box if the majority of workers drew a box that could be matched to it. Figs. 3.4a,3.5a,3.5c show results for different lesioned methods. In each method name, the tag *worker* means that a worker skill and image difficulty model was used, the tag *online* means that online crowdsourcing was used (with parameter  $\tau_\epsilon = .005$ , unless a different number appears in the method name), the tag *cvnaive* means that a naive method to incorporate computer vision was used (by treating computer vision like a human worker, see Section 3.4), and the tag *cv* means that computer vision probabilities described in Section 3.6-3.7,3.8 were used.



**Figure 3.5: Crowdsourcing Multi-Object Bounding Box and Part Annotations:** (a) Our full model prob-worker-cvonline-0.02 obtains slightly lower error than majority-vote while using only 1.97 workers per image. (b) Histogram of the number of human annotators per image. (c) The worker skill model (prob-worker) led to 10% reduction in error over the majority-vote baseline, and the online model cut annotation time roughly in half. (d) Histogram of the number of human annotators per part.

### Binary Annotation

We collected 3 datasets (scorpions, beakers, and cardigan sweaters) which we believe to be representative of the way datasets like ImageNet (Deng, Dong, et al., 2009) and CUB-200-2011 (Wah, Branson, Welinder, et al., 2011) were collected. For each

category, we collected 4000 Flickr images by searching for the category name. 15 MTurkers per image were asked to filter search results. We obtained ground truth labels by carefully annotating images ourselves. Fig. 3.4a summarizes performance for the scorpion category (which is typical, see supplementary material for results on more categories), whereas Fig. 3.4c shows qualitative examples.

The full model prob-worker-cvonline-0.02 obtained results as good as typical baselines with 15 workers (majority-vote and prob) using only 1.37 workers per image on average. The method prob-online corresponds to the online crowdsourcing method of Welinder et al. (Welinder and Perona, 2010), which used 5.1 workers per image and resulted in an error of 0.045; our full method prob-worker-cvonline-0.005 obtained lower error 0.041 with only 1.93 workers per image. We see that incorporating a worker skill model reduced converged error by about 33% (comparing prob-worker to majority-vote or prob). Adding online crowdsourcing roughly halved the number of annotations required to obtain comparable error (comparing prob-worker-online vs. prob-worker). Adding computer vision reduced the number of annotations per image by an additional factor of 2.4 with comparable error (comparing prob-worker-cvonline-0.005 to prob-worker-online). It also reduced annotations by a factor of 1.8 compared to the naive method of using computer vision (prob-worker-cvnaive-online), showing that using computer vision confidence estimates is useful. Interestingly, in Fig. 3.4b we see that adding computer vision allowed many images to be predicted confidently using no worker labels. Lastly, comparing prob-worker-cvonline-0.02 to prob-worker-cvonline-0.005, which resulted in errors of 0.051 and 0.041, respectively, and 1.37 vs. 1.93 workers per image, we see that the error tolerance parameter  $\tau_\epsilon$  offers an intuitive parameter to tradeoff annotation time and quality.

### **Bounding Box Annotation**

To evaluate bounding box annotation, we used a 1448 image subset of the Caltech Roadside Pedestrian dataset (Hall and Perona, 2015). The dataset is challenging, because it contains images of pedestrians in the wild; some images contain no pedestrians, others contain many, pedestrians are often walking next to each other causing overlapping bounding boxes, and some pedestrians are far away and less than 10 pixels. We obtained ground truth annotations and 7 MTurk annotations per image from the creators of the dataset. We incur error for all false positives and negatives using a .5 IOU overlap criterion.



Figure 3.6: Binary classification sequences. The image on the left represents a best case scenario where the computer vision is able to confidently label the image. The image in the center represents an average annotation situation where the computer vision label and one worker label is needed to confidently label the image. The image on the right (which is not a scorpion) represents a difficult case in which many workers disagreed on the label.

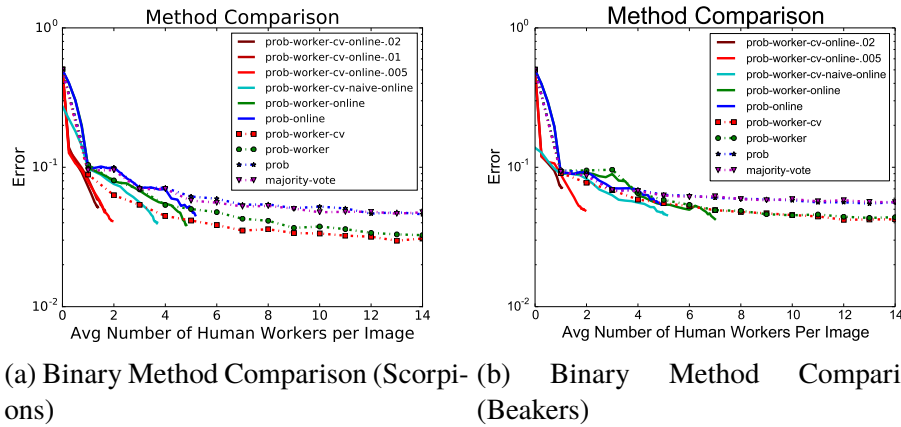


Figure 3.7: **Crowdsourcing Binary Classification Annotations:** We ran binary classification experiments on two different datasets: scorpions and beakers, which were selected because they are two different types of objects that pose different challenges: scorpions are natural objects, with some grossly different objects in search results (the band scorpions, the video game character, the motorcycle) and some very related objects (scorpion spiders are spiders that resemble scorpions). Beakers are man-made objects, which are often not the subject of the photo (and thus not centered and very small in the image), and many people mistakenly tag images of flasks and granulated cylinders as beakers. **(a) Results on the Scorpion Dataset:** Our full model prob-worker-cvonline-0.02 obtains results as good as typical baselines with 15 workers (majority-vote and prob) using only 1.37 workers per image on average. **(b) Results on the Beakers Dataset:** Similar trends occur for both beakers and scorpions

In Fig. 3.5a, we see that the full model `prob-worker-cvonline-0.02` obtained slightly lower error than majority-vote while using only 1.97 workers per image. This is encouraging, given that most publicly available crowdsourcing tools for bounding box annotation use simple crowdsourcing methods. Incorporating a probabilistic model (comparing prob to majority-vote) reduced the error by a factor of 2, demonstrating that it is useful to account for probabilities of false positive and false negative boxes, and precision in drawing box boundaries. Online crowdsourcing reduced the number of required workers per image by a factor of 1.7 without increasing error (comparing `prob-worker-online` to `prob-worker`), while adding computer vision (method `prob-worker-online-.005`) reduced annotation by an additional 29%. Examining Fig. 3.5b, we see that computer vision allowed many images to be confidently annotated with a single human worker. The naive computer vision method `prob-worker-cvnaive-online` performed as well as our more complicated method.

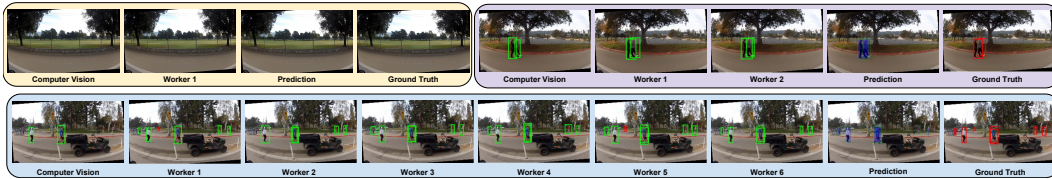


Figure 3.8: Bounding box annotation sequences. The top left annotation sequence highlights a good case where only the computer vision system and one human are needed to finish the image. The top right annotation sequence highlights the average case where two workers and the computer vision system are needed to finish the image. The bottom row highlights a difficult annotation sequence where workers disagree on the number of instances, forcing the image to remain unfinished longer than usual.

### Part Annotation

To evaluate part keypoint annotation, we used the 1000 image subset of the NABirds dataset (Van Horn et al., 2015), for which a detailed analysis comparing experts to MTurkers was performed in (Van Horn et al., 2015). This subset contained 10 MTurker labels per image of 11 semantic keypoint locations as well as expert part labels. Although our algorithm processed each part independently, we report error averaged over all 11 parts, using the loss defined in Section 3.7. We did not implement a computer vision algorithm for parts; however, a variant of our algorithm (`prob-worker-online`) was used by the creators of the dataset to collect its published part annotations (11 parts on 55,000 images), using only 2.3 worker annotations per part on average.

Simulated results on the 1000 image subset are shown in Fig. 3.5c. We see that the worker skill model (prob-worker) led to 10% reduction in error over the majority-vote baseline, and online model cut annotation time roughly in half, with most parts finishing with 2 worker clicks (Fig.3.4b)

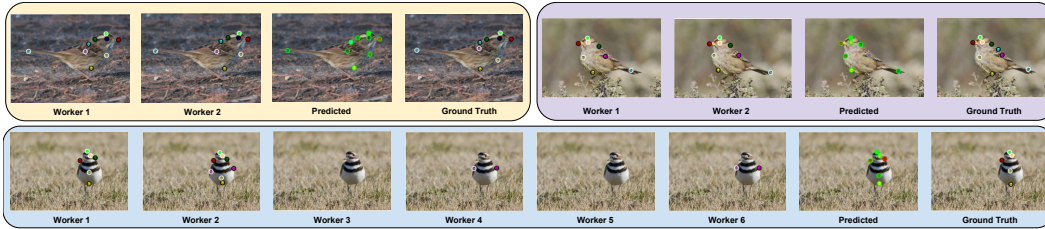


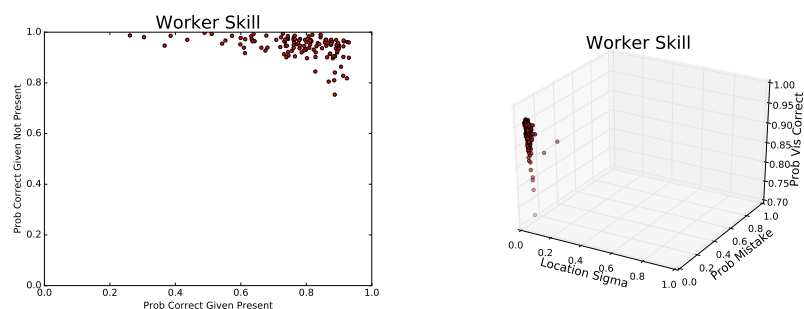
Figure 3.9: Part annotation sequences. The two sequences on the top row are the common situation where the responses from two workers correlate well and are enough for the system to mark the images as finished. The annotation sequence on the bottom row highlights a difficult situation where workers toggle back and forth on the visibility of the wings, forcing the image to remain unfinished for longer than usual. This toggling behavior can be attributed to task ambiguity and/or insufficient instructions.

### Worker Skills

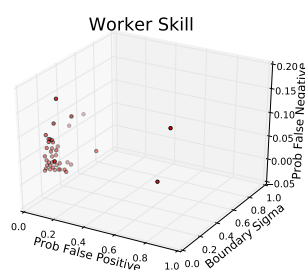
One bonus of our algorithm is that it predicts the skill of each worker according to a small number of semantically interpretable features. These could be used for blocking spammers, giving bonuses to good workers, or debugging ambiguities in the annotation task.

### Discussion and Failure Cases

All crowdsourcing methods resulted in some degree of error when crowd labels converged to something different than expert labels. The most common reason was ambiguous images. For example, most MTurkers incorrectly thought scorpion spiders (a type of spider resembling scorpions) were actual scorpions. Visibility of a part annotation can become ambiguous as an object rotates from frontal to rear view. However, all variants of our method (with and without computer vision, with and without online crowdsourcing) resulted in higher quality annotations than majority vote (which is commonly used for many computer vision datasets). Improvement in annotation quality came primarily from modeling worker skill. Online crowdsourcing can increase annotation errors; however, it does so with an interpretable parameter for trading off annotation time and error. Computer vision also reduces annotation time, with greater gains coming as dataset size increases. However, we



(a) Worker Skills for Binary Classification (b) Worker Skills for Part Annotations



(c) Worker Skills for Bounding Box Annotations

Figure 3.10: **Inferred Worker Skills:** These plots easily surface worker capabilities and can inform researchers on the difficulty of their task and help debug ambiguous tasks. **(a)** On the scorpion dataset (binary classification), most workers on this task can tell fairly accurately when a scorpion is not present; however, there is a large spread in worker miss rate, possibly due to less careful workers in picking out smaller or more ambiguous objects. **(b)** On the bird dataset (part annotation), workers tend to be highly accurate when annotating parts. Visibility issues are often due to left/right part mistakes or when a part is partially (self) occluded. **(c)** On the pedestrian dataset (bounding box annotation), workers tend to not hallucinate people (low false positive), however there is a chance that they miss a (probably small) instance.



note that in some cases, adding computer vision in the loop might be inappropriate for research datasets due to bias toward certain algorithms. We allow it to be toggled on or off in our source code.

### 3.10 Conclusion

In this work, we introduced crowdsourcing algorithms and online tools for collecting binary, part, and bounding box annotations. We showed that each component of the system—a worker skill / image difficulty model, an online stoppage criterion for collecting a variable number of annotations per image, and integration of computer vision in the loop—led to significant reductions in annotation time and/or annotation error for each type of annotation. In future work, we plan to extend the approach to other types of annotation, like segmentation and video, use inferred worker skill parameters to block spammers, choose which worker should annotate an image, and incorporate active learning criteria to choose which images to annotate next or choose between different types of user interfaces.

### Acknowledgments

This paper was inspired by work from and earlier collaborations with Peter Welinder and Boris Babenko. Much thanks to Pall Gunnarsson for helping to develop an early version of the method. Thank you to David Hall for supplying data for bounding box experiments. This work was supported by a Google Focused Research Award and Office of Naval Research MURI N000141010933.

### References

- Andriluka, Mykhaylo et al. (2014). “2D Human Pose Estimation: New Benchmark and State of the Art Analysis”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Biswas, Arijit and Devi Parikh (2013). “Simultaneous active learning of classifiers & attributes via relative feedback”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 644–651.
- Branson, Steve et al. (2010). “Visual recognition with humans in the loop”. In: *European Conference on Computer Vision*. Springer, pp. 438–451.
- Carpenter, Bob (2008). “Multilevel bayesian models of categorical data annotation”. In: *Unpublished manuscript*.
- Chilton, Lydia B et al. (2013). “Cascade: Crowdsourcing taxonomy creation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pp. 1999–2008.

- Dalvi, Nilesh et al. (2013). “Aggregating crowdsourced binary ratings”. In: *Proceedings of the 22nd international conference on World Wide Web*. ACM, pp. 285–294.
- Dawid, Alexander Philip and Allan M Skene (1979). “Maximum likelihood estimation of observer error-rates using the EM algorithm”. In: *Applied statistics*, pp. 20–28.
- Deng, Jia, Wei Dong, et al. (2009). “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pp. 248–255.
- Deng, Jia, Jonathan Krause, and Li Fei-Fei (2013). “Fine-grained crowdsourcing for fine-grained recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587.
- Deng, Jia, Olga Russakovsky, et al. (2014). “Scalable multi-label annotation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pp. 3099–3102.
- Dutt Jain, Suyog and Kristen Grauman (2013). “Predicting sufficient annotation strength for interactive foreground segmentation”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1313–1320.
- Erlenkotter, Donald (1978). “A dual-based procedure for uncapacitated facility location”. In: *Operations Research* 26.6, pp. 992–1009.
- Gao, Chao and Dengyong Zhou (2013). “Minimax optimal convergence rates for estimating ground truth from crowdsourced labels”. In: *arXiv preprint arXiv:1310.5764*.
- Ghosh, Arpita, Satyen Kale, and Preston McAfee (2011). “Who moderates the moderators?: crowdsourcing abuse detection in user-generated content”. In: *Proceedings of the 12th ACM conference on Electronic commerce*. ACM, pp. 167–176.
- Griffin, Gregory, Alex Holub, and Pietro Perona (2007). “Caltech-256 object category dataset”. In:
- Gurari, Danna et al. (2015). “How to collect segmentations for biomedical images? A benchmark evaluating the performance of experts, crowdsourced non-experts, and algorithms”. In: *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, pp. 1169–1176.
- Hall, David and Pietro Perona (2015). “Fine-grained classification of pedestrians in video: Benchmark and state of the art”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5482–5491.
- He, Kaiming et al. (2015). “Deep residual learning for image recognition”. In: *arXiv preprint arXiv:1512.03385*.

- Hua, Gang et al. (2013). “Collaborative active learning of a kernel machine ensemble for recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1209–1216.
- Jain, Suyog Dutt and Kristen Grauman (2016). “Active Image Segmentation Propagation”. In: CVPR.
- Jin, Rong and Zoubin Ghahramani (2002). “Learning with multiple labels”. In: *Advances in neural information processing systems*, pp. 897–904.
- Karger, David R, Sewoong Oh, and Devavrat Shah (2011). “Iterative learning for reliable crowdsourcing systems”. In: *Advances in neural information processing systems*, pp. 1953–1961.
- (2013). “Efficient crowdsourcing for multi-class labeling”. In: *ACM SIGMETRICS Performance Evaluation Review* 41.1, pp. 81–92.
- Kazemzadeh, Sahar et al. (2014). “ReferItGame: Referring to Objects in Photographs of Natural Scenes.” In: *EMNLP*, pp. 787–798.
- Khodabandeh, Mehran et al. (2015). “Discovering human interactions in videos with limited data labeling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 9–18.
- Kovashka, Adriana, Sudheendra Vijayanarasimhan, and Kristen Grauman (2011). “Actively selecting annotations among objects and attributes”. In: *2011 International Conference on Computer Vision*. IEEE, pp. 1403–1410.
- Kovashka, A. et al. (2016). “Crowdsourcing in Computer Vision”. In: *ArXiv e-prints*. arXiv: 1611.02145 [cs.CV]. URL: <https://arxiv.org/abs/1611.02145>.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks.” In: *NIPS*.
- Lad, Shrenik and Devi Parikh (2014). “Interactively guiding semi-supervised clustering via attribute-based explanations”. In: *European Conference on Computer Vision*. Springer, pp. 333–349.
- Larlus, Diane et al. (2014). “Generating Gold Questions for Difficult Visual Recognition Tasks”. In:
- Li, Hongwei and Bin Yu (2014). “Error rate bounds and iterative weighted majority voting for crowdsourcing”. In: *arXiv preprint arXiv:1411.4086*.
- Lin, Tsung-Yi et al. (2014). “Microsoft COCO: Common objects in context”. In: *ECCV*.
- Liu, Qiang, Jian Peng, and Alexander T Ihler (2012). “Variational inference for crowdsourcing”. In: *Advances in Neural Information Processing Systems*, pp. 692–700.

- Long, Chengjiang and Gang Hua (2015). “Multi-class multi-annotator active learning with robust Gaussian Process for visual recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2839–2847.
- Long, Chengjiang, Gang Hua, and Ashish Kapoor (2013). “Active visual recognition with expertise estimation in crowdsourcing”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3000–3007.
- Ok, Jungseul et al. (2016). “Optimality of Belief Propagation for Crowdsourced Classification”. In: *arXiv preprint arXiv:1602.03619*.
- Parkash, Amar and Devi Parikh (2012). “Attributes for classifier feedback”. In: *European Conference on Computer Vision*. Springer, pp. 354–368.
- Patterson, Genevieve, Grant Van Horn<sup>2</sup> Serge Belongie, and Pietro Perona<sup>2</sup> James Hays (2015). “Tropel: Crowdsourcing Detectors with Minimal Training”. In: *Third AAAI Conference on Human Computation and Crowdsourcing*.
- Platt, John et al. (1999). “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods”. In: *Advances in large margin classifiers* 10.3, pp. 61–74.
- Raykar, Vikas C et al. (2010). “Learning from crowds”. In: *Journal of Machine Learning Research* 11.Apr, pp. 1297–1322.
- Rubinstein, Michael, Ce Liu, and William T Freeman (2012). “Annotation propagation in large image databases via dense image correspondence”. In: *European Conference on Computer Vision*. Springer, pp. 85–99.
- Russakovsky, Olga, Li-Jia Li, and Li Fei-Fei (2015). “Best of both worlds: human-machine collaboration for object annotation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2121–2131.
- Russell, Bryan C et al. (2008). “LabelMe: a database and web-based tool for image annotation”. In: *International journal of computer vision* 77.1-3, pp. 157–173.
- Shah, Nihar B, Sivaraman Balakrishnan, and Martin J Wainwright (2016). “A permutation-based model for crowd labeling: Optimal estimation and robustness”. In: *arXiv preprint arXiv:1606.09632*.
- Shah, Nihar Bhadresh and Denny Zhou (2015). “Double or nothing: Multiplicative incentive mechanisms for crowdsourcing”. In: *Advances in Neural Information Processing Systems*, pp. 1–9.
- Shankar Nagaraja, Naveen, Frank R Schmidt, and Thomas Brox (2015). “Video Segmentation with Just a Few Strokes”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3235–3243.
- Sheng, Victor S, Foster Provost, and Panagiotis G Ipeirotis (2008). “Get another label? improving data quality and data mining using multiple, noisy labelers”. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 614–622.

- Siddiquie, Behjat and Abhinav Gupta (2010). “Beyond active noun tagging: Modeling contextual interactions for multi-class active learning”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, pp. 2979–2986.
- Smyth, Padhraic et al. (1995). “Inferring ground truth from subjective labelling of venus images”. In:
- Snow, Rion et al. (2008). “Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks”. In: *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pp. 254–263.
- Sorokin, Alexander and David Forsyth (2008). “Utility data annotation with amazon mechanical turk”. In: *Urbana* 51.61, p. 820.
- Su, Hao, Jia Deng, and Li Fei-Fei (2012). “Crowdsourcing annotations for visual object detection”. In: *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*.
- Szegedy, Christian, Wei Liu, et al. (2015). “Going deeper with convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9.
- Szegedy, Christian, Scott Reed, et al. (2014). “Scalable, high-quality object detection”. In: *arXiv preprint arXiv:1412.1441*.
- Tian, Tian and Jun Zhu (2015). “Max-margin majority voting for learning from crowds”. In: *Advances in Neural Information Processing Systems*, pp. 1621–1629.
- Van Horn, Grant et al. (2015). “Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604. DOI: 10.1109/CVPR.2015.7298658.
- Vijayanarasimhan, Sudheendra and Kristen Grauman (2009a). “Multi-level active prediction of useful image annotations for recognition”. In: *Advances in Neural Information Processing Systems*, pp. 1705–1712.
- (2009b). “What’s it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pp. 2262–2269.
- Vittayakorn, Sirion and James Hays (2011). “Quality Assessment for Crowdsourced Object Annotations.” In: *BMVC*, pp. 1–11.
- Von Ahn, Luis and Laura Dabbish (2004). “Labeling images with a computer game”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 319–326.

- Von Ahn, Luis and Laura Dabbish (2005). “ESP: Labeling Images with a Computer Game.” In: *AAAI spring symposium: Knowledge collection from volunteer contributors*. Vol. 2.
- Vondrick, Carl, Donald Patterson, and Deva Ramanan (2013). “Efficiently scaling up crowdsourced video annotation”. In: *International Journal of Computer Vision* 101.1, pp. 184–204.
- Wah, Catherine, Steve Branson, Pietro Perona, et al. (2011). “Multiclass recognition and part localization with humans in the loop”. In: *2011 International Conference on Computer Vision*. IEEE, pp. 2524–2531.
- Wah, Catherine, Steve Branson, Peter Welinder, et al. (2011). “The caltech-ucsd birds-200-2011 dataset”. In:
- Wah, Catherine, Grant Van Horn, et al. (2014). “Similarity comparisons for interactive fine-grained categorization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 859–866.
- Wang, Jing, Panagiotis G Ipeirotis, and Foster Provost (2013). “Quality-based pricing for crowdsourced workers”. In:
- Welinder, Peter, Steve Branson, et al. (2010). “The multidimensional wisdom of crowds”. In: *Advances in neural information processing systems*, pp. 2424–2432.
- Welinder, Peter and Pietro Perona (2010). “Online crowdsourcing: rating annotators and obtaining cost-effective labels”. In:
- Whitehill, Jacob et al. (2009). “Whose vote should count more: Optimal integration of labels from labelers of unknown expertise”. In: *Advances in neural information processing systems*, pp. 2035–2043.
- Wilber, Michael J, Iljung S Kwak, and Serge J Belongie (2014). “Cost-effective hits for relative similarity comparisons”. In: *Second AAAI Conference on Human Computation and Crowdsourcing*.
- Yao, Angela et al. (2012). “Interactive object detection”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 3242–3249.
- Zhang, Chicheng and Kamalika Chaudhuri (2015). “Active learning from weak and strong labelers”. In: *Advances in Neural Information Processing Systems*, pp. 703–711.
- Zhang, Yuchen et al. (2014). “Spectral methods meet EM: A provably optimal algorithm for crowdsourcing”. In: *Advances in neural information processing systems*, pp. 1260–1268.
- Zhou, Dengyong et al. (2015). “Regularized minimax conditional entropy for crowdsourcing”. In: *arXiv preprint arXiv:1503.07240*.
- Zhou, Denny et al. (2012). “Learning from the wisdom of crowds by minimax entropy”. In: *Advances in Neural Information Processing Systems*, pp. 2195–2203.

*Chapter 4*

## LEAN MULTICLASS CROWDSOURCING

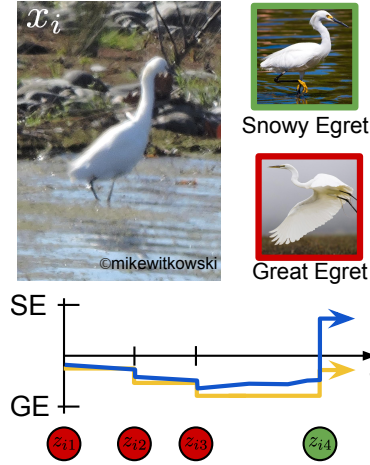
Van Horn, Grant et al. (2018). “Lean Multiclass Crowdsourcing”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT. DOI: 10.1109/cvpr.2018.00287.

**4.1 Abstract**

We introduce a method for efficiently crowdsourcing multiclass annotations in challenging, real world image datasets. Our method is designed to minimize the number of human annotations that are necessary to achieve a desired level of confidence on class labels. It is based on combining models of worker behavior with computer vision. Our method is general: it can handle a large number of classes, worker labels that come from a taxonomy rather than a flat list, and can model the dependence of labels when workers can see a history of previous annotations. Our method may be used as a drop-in replacement for the majority vote algorithms used in online crowdsourcing services that aggregate multiple human annotations into a final consolidated label. In experiments conducted on two real-life applications, we find that our method can reduce the number of required annotations by as much as a factor of 5.4 and can reduce the residual annotation error by up to 90% when compared with majority voting. Furthermore, the online risk estimates of the models may be used to sort the annotated collection and minimize subsequent expert review effort.

**4.2 Introduction**

Multiclass crowdsourcing is emerging as an important technique in science and industry. For example, a growing number of websites support sharing observations (photographs) of specimens from the natural world and facilitate collaborative, community-driven identification of those observations. Websites such as iNaturalist, eBird, Mushroom Observer, HerpMapper, and LepSnap accumulate large collections of images and identifications, often using majority voting to produce the final species label. Ultimately, this information is aggregated into datasets (e.g. GBIF (Ueda, 2017)) that enable global biodiversity studies (Sullivan et al., 2014). Thus, the label accuracy of these datasets can have a direct impact on science, conservation, and policy. Thanks to the recent dramatic improvements in our



**Figure 4.1: iNaturalist Community Identification.** A user uploads image  $x_i$  (top-left) with an initial species prediction  $z_{i1} = \text{Great Egret (GE)}$ , one out of 1.5k North American bird species. Later, two additional users (potentially alerted that a GE has been spotted) come along and, after inspecting the image *and* the previous identifications, contribute their subjective identifications of the bird species  $z_{i2} = \text{GE}$  and  $z_{i3} = \text{GE}$ , agreeing with the uploader. Finally, a fourth user provides a different identification  $z_{i4} = \text{Snowy Egret (SE)}$ . In the plot below the images, two models (red, green) integrate the information differently, with the y axis representing likelihood of SE vs. GE. Majority voting (yellow arrow) simply tallies the vote, and GE is the chosen answer after four votes. Our model (blue arrow) continuously analyzes the users' skills across *other* observations and is therefore capable of updating the likelihood of the predicted label much more frequently. Knowing that the fourth user is highly skilled on these taxa, our model overrides previous users and predicts SE. The underlying ground truth answer is indeed SE. In this work, we design and compare several models that estimate user skill and use it to weigh votes appropriately. (View on iNat: <https://www.inaturalist.org/observations/4599411>)

field (Krizhevsky, Sutskever, and Hinton, 2012; He et al., 2015; Szegedy et al., 2016; Huang et al., 2017), observations collected by these websites can be used to train classification services (e.g. see [merlin.allaboutbirds.org](http://merlin.allaboutbirds.org) and [inaturalist.org](http://inaturalist.org)), helping novices label their observations. The result is an even larger collection of observations, but with potentially noisier labels, as the number of people taking photos and submitting observations far outpaces the speed at which experts can verify them. The benefits of a simple algorithm like majority vote are lost when the skill of the people contributing labels is uncertain. Thus, there is need for improved methods to integrate multiple identifications into a final label.

Figure 4.1 shows a real example of a user's observation on iNaturalist, a sequence of identifications from the community, and how the current species label is computed



using majority voting. The structure of these interactions present three challenges that have not been tackled by prior work on combining multiclass annotations (Denny Zhou et al., 2012; Karger, Oh, and D. Shah, 2013; Vempaty, L. R. Varshney, and P. K. Varshney, 2014; Dengyong Zhou et al., 2014; Y. Zhang et al., 2014; Tian and Zhu, 2015; J. Zhang et al., 2016): (1) iNaturalist has a tree structured taxonomy of labels rather than a flat list, allowing users to provide labels at varying depths of the taxonomy depending on their confidence; (2) identifiers get to see the history of previous identifications for an observation, so their identification is *not* independent of previous identifiers; and (3) the number of species under consideration is huge, currently at  $\sim 130,000$  but potentially reaching 8M (Mora et al., 2011).

We propose a new method for aggregating multiple multiclass labels. Our method is based on models of worker behavior and can replace majority vote in websites like iNaturalist, and in more traditional data labeling services (e.g. Amazon Mechanical Turk). We show that our models are more accurate than majority voting (reducing error by 90% on data from iNaturalist), and when combined with a computer vision system, can drastically reduce the number of labels required per image (e.g. by a factor of 5.4 on crowdsourced data). Our main contribution is a method for *multiclass annotation* tasks that (1) can be used in online crowdsourcing, (2) can handle large numbers of classes, (3) can handle a taxonomy of labels allowing workers to respond at coarser levels than leaf nodes, and (4) can handle mutually dependent worker labels.

### 4.3 Related Work

Kovashka et al. (Kovashka et al., 2016) provide a thorough review of crowdsourcing techniques for computer vision. The Dawid-Skene (DS) model (Dawid and Skene, 1979) is the standard probabilistic model for multiclass label inference from multiple annotations. That model assumes each worker has a latent confusion matrix that captures the probability of annotating a class correctly (the diagonal entries) and the probability of confusing two classes (the off diagonal entries). The DS model iteratively infers the reliability of each worker and updates the belief of the true labels, using Expectation-Maximization as the inference algorithm. Alternate inference algorithms for the DS model are based on spectral methods (Ghosh, Kale, and McAfee, 2011; Dalvi et al., 2013; Karger, Oh, and D. Shah, 2011; Karger, Oh, and D. Shah, 2013; Karger, Oh, and D. Shah, 2014; Y. Zhang et al., 2014), belief propagation (Liu, Peng, and Ihler, 2012; Ok et al., 2016), expectation maximization (Liu, Peng, and Ihler, 2012; Y. Zhang et al., 2014), maximum entropy (Denny Zhou et al.,

2012; Dengyong Zhou et al., 2014), weighted majority voting (Littlestone and Warmuth, 1994; Li, Yu, and Dengyong Zhou, 2013), and max-margin (Tian and Zhu, 2015). Alternatives to the DS model have also been proposed (Smyth et al., 1995; Jin and Ghahramani, 2002; Whitehill et al., 2009; Welinder et al., 2010; Raykar et al., 2010; Tang and Lease, 2011; Kamar, Hacker, and Horvitz, 2012; J. Zhang et al., 2016; Branson, Van Horn, and Perona, 2017; Chen et al., 2017). Further work based on active learning tackles noisy labelers (Long, Hua, and Kapoor, 2013) and task allocation to minimize the monetary cost of dataset construction (Karger, Oh, and D. Shah, 2013; Karger, Oh, and D. Shah, 2014; N. B. Shah and Denny Zhou, 2015).

Multiclass tasks, as opposed to binary tasks, are explored by (Denny Zhou et al., 2012; Karger, Oh, and D. Shah, 2013; Vempaty, L. R. Varshney, and P. K. Varshney, 2014; Dengyong Zhou et al., 2014; Y. Zhang et al., 2014; Tian and Zhu, 2015; J. Zhang et al., 2016; Chen et al., 2017). Zhou et al. use entropy maximization to model both worker confusions and task difficulties for multiclass Denny Zhou et al., 2012 and ordinal (Dengyong Zhou et al., 2014) data. Similarly, Chen et al. (Chen et al., 2017) use max-margin techniques to further improve results for ordinal tasks. Karger et al. (Karger, Oh, and D. Shah, 2013) use an iterative algorithm by converting  $k$ -class tasks into  $k - 1$  binary tasks but make assumptions on the number of items and workers. Vempaty et al. (Vempaty, L. R. Varshney, and P. K. Varshney, 2014) also convert  $k$ -class tasks into binary tasks, but take a coding theoretic approach to estimate labels. Zhang et al. (Y. Zhang et al., 2014) use spectral methods to initialize the EM inference algorithm of the Dawid-Skene model, while Tian et al. (Tian and Zhu, 2015) fuse a max-margin estimator and the Dawid-Skene model. Zhang et al. (J. Zhang et al., 2016) create probabilistic features for each item and use a clustering algorithm to assign them their final labels, however they do not produce an estimate of worker skill. All of the previous approaches assume that annotations are independent. We differentiate our work by handling both independent *and* dependent annotations collected by sites like iNaturalist. Furthermore, we explore the challenges of “large-scale” multiclass task modeling where the number of classes is nearly  $10\times$  larger than the prior art has explored. Our work also handles taxonomic modeling of the classes and non-leaf node worker annotations. See Table 4.2 for a performance comparison of our model to prior art.

Final label quality between independent and dependent crowdsourcing tasks is studied by Little et al. (Little et al., 2010), but without modeling workers. The

work of Branson et al. (Branson, Van Horn, and Perona, 2017) is the closest to ours, as we adapt their framework to multiclass annotation, which they did not investigate. Furthermore, we explore taxonomic multiclass annotations to reduce the number of parameters. Additionally, we develop models that do not depend on the assumption that worker annotations are independent, and we are thus able to handle mutually dependent annotations where each worker can see previous labels.

#### 4.4 Multiclass Online Crowdsourcing

Given a set of worker annotations  $Z$  for a dataset of images  $X$ , the probabilistic framework of Branson et al. (Branson, Van Horn, and Perona, 2017) jointly models worker skill  $W$ , image difficulty  $D$ , ground truth labels  $Y$ , and computer vision system parameters  $\theta$ . A tiered prior system is used to make the system more robust by regularizing the per worker skill and image difficulty priors. Alternating maximization is used for parameter estimation. The Bayesian risk  $\mathcal{R}(\bar{y}_i)$  (see Eq.1 from (Branson, Van Horn, and Perona, 2017)) can be computed for each predicted label, providing an intuitive online stopping criteria (i.e. the model can “retire” images as soon as their risk is below a threshold  $\tau_\epsilon$ ). In this work, we extend this framework by implementing multiple models of worker skill for the task of multiclass annotation for independent and dependent worker labels. For our experiments, we removed the image difficulty part of the framework and focused solely on modeling workers and their labels. Section 4.4 constructs worker skill models when the labels  $Z$  are independent, and Section 4.4 constructs worker skill models when the labels  $Z$  are dependent.

##### Independent Labels

Let  $x_i$  be the  $i$ th image, which contains an object with class label  $y_i \in \{1, \dots, C\}$  (e.g., species). Suppose a set of workers  $\mathcal{W}_i$  independently specify their guess at the class of image  $i$ , such that for each  $j \in \mathcal{W}_i$ ,  $z_{ij}$  is worker  $j$ ’s guess at  $y_i$ . In this situation, identifiers from Figure 4.1 would not get to observe preceding users’ guesses. Let  $w_j$  be some set of parameters encoding worker  $j$ ’s skill at predicting classes. In this notation, if the class  $y_i$  is unknown, we can estimate the probability of each possible class given the set  $Z_i = \{z_{ij}\}_{j \in \mathcal{W}_i}$  of worker guesses:

$$p(y_i|Z_i) = \frac{p(y_i) \prod_{j \in \mathcal{W}_i} p(z_{ij}|y_i, w_j)}{\sum_{y=1}^C p(y) \prod_{j \in \mathcal{W}_i} p(z_{ij}|y, w_j)} \quad (4.1)$$

where  $p(y_i)$  is the prior class probability and  $p(z_{ij}|y_i, w_j)$  is a model of imperfect human guesses. The following sections discuss possible models for  $p(z_{ij}|y_i, w_j)$ ,

Name	Interpretation	Model	Expression	# Params	# Params For Birds
<b>Flat Single Binomial</b>	Probability of being correct is the same for all species	$z = y$ is binomial with the same parameters regardless of $y$	$p(z y) = \begin{cases} m & \text{if } z = y \\ (1 - m)p(z) & \text{otherwise} \end{cases}$	1	1
<b>Flat Per Class Binomial</b>	Probability of being correct for each species separately	For each value $y = c$ , $z = y$ is binomial	$p(z y) = \begin{cases} M(y) & \text{if } z = y \\ (1 - M(y))p(z) & \text{otherwise} \end{cases}$	$C$	1,572
<b>Flat Per Class Multinomial</b>	Confusion probability over each pair of species	For each value $y = c$ , $z$ is multinomial	$p(z y) = M(y, z)$	$C^2$	2,471,184
<b>Taxonomic Single Binomial</b>	Probability of being correct is the same for each species in a genus	$z^l = y^l   z^{l-1} = y^{l-1}$ is binomial with the same parameters regardless of $y^l$	$p(z y) = \prod_l p(z^l   y^l)$ $p(z^l   y^l) = \begin{cases} m_{y^{l-1}} & \text{if } z^l = y^l \\ (1 - m_{y^{l-1}})p(z^l) & \text{otherwise} \end{cases}$	$ N  - C$	383
<b>Taxonomic Per Class Binomial</b>	Probability of being correct for each species separately	For each value $y^l = c$ , $z^l = y^l   z^{l-1} = y^{l-1}$ is binomial	$p(z y) = \prod_l p(z^l   y^l)$ $p(z^l   y^l) = \begin{cases} M_{y^{l-1}}(y^l) & \text{if } z^l = y^l \\ (1 - M_{y^{l-1}}(y^l))p(z) & \text{otherwise} \end{cases}$	$ N $	1955
<b>Taxonomic Per Class Multinomial</b>	Confusion probability for each pair of species in a genus	For each value $y^l = c$ , $z^l   z^{l-1} = y^{l-1}$ is multinomial	$p(z y) = \prod_l p(z^l   y^l)$ $p(z^l   y^l) = M_{y^{l-1}}(y^l, z^l)$	$\sum_{n \in N}  \text{children}(n) ^2$	22,472

Table 4.1: Different options for modeling worker skill given a taxonomy of classes.  $N$  is the set of nodes in the taxonomic tree,  $C$  is the number of leaf nodes (i.e. class labels). The last column shows the number of resulting parameters when modeling the 1,572 species of North American birds and their taxonomy from the iNaturalist database, *for a single worker*. Multinomial models have significantly more parameters but can model commonly confused classes. Taxonomic methods have the benefit of supporting non-species-level human responses, modeling skill at certain taxa, and reducing the number of parameters for multinomial models.

which are also summarized in Table 4.1.

### Flat Models

**Flat Single Binomial:** One simple way to model worker skills is with a single parameter that captures the worker's probability of providing a correct answer, regardless of the class label. We assume that the probability of a worker being correct  $m_j$  follows a Bernoulli distribution, with other responses having probability proportional to class priors:

$$p(z_{ij}|y_i, w_j) = \begin{cases} m_j & \text{if } z_{ij} = y_i \\ (1 - m_j)p(z_{ij}) & \text{otherwise} \end{cases} \quad (4.2)$$

To prevent over fitting in low data situations, we place a beta prior  $\text{Beta}(n_\beta p^c, n_\beta(1 - p^c))$  on  $m_j$ , where  $n_\beta$  is the strength of the prior.  $p^c$  represents the probability of any worker providing a correct label, and is estimated by pooling all worker annotations together. We also place a beta prior  $\text{Beta}(n_\beta p, n_\beta(1 - p))$  on  $p^c$ , with  $p$  acting as our prior belief on worker performance. Estimating the worker skills is done by counting the number of times their response agrees with the predicted label, weighted by the prior strength:

$$m_j = \frac{n_\beta p^c + \sum_{i \in \mathcal{I}_j} 1[z_{ij} = \bar{y}_i, |\mathcal{W}_i| > 1] - 1}{n_\beta + \sum_{i \in \mathcal{I}_j} 1[\bar{y}_i, |\mathcal{W}_i| > 1] - 2} \quad (4.3)$$

where  $1[\cdot]$  is the indicator function,  $\mathcal{I}_j$  are the images labeled by worker  $j$ , and  $\bar{y}_i$  is our current label prediction for image  $i$ . The pooled prior  $p^c$  is estimated similarly.

**Flat Per Class Binomial:** Rather than learning a single skill parameter  $m$  across all classes, we can learn a separate binomial model for each value of  $y$ , resulting in a skill vector  $M_j$  for each worker:

$$p(z_{ij}|y_i, w_j) = \begin{cases} M_j(y_i) & \text{if } z_{ij} = y_i \\ (1 - M_j(y_i))p(z_{ij}) & \text{otherwise} \end{cases} \quad (4.4)$$

Similar to the single binomial model, we employ a tiered prior system by adding a per class beta prior  $\text{Beta}(n_\beta p^y, n_\beta(1 - p^y))$  on  $M_j(y)$ . We place a generic beta prior  $\text{Beta}(n_\beta p, n_\beta(1 - p))$  on  $p^y$  to encode our prior belief that a worker is correct on any class. Estimating the worker skill parameters  $M_j(y)$  and the pooled priors  $p^y$  for class  $y$  is done in the same way as the single binomial model.

**Flat Per Class Multinomial:** A more sophisticated model of  $p(z_{ij}|y_i, w_j)$  could assume  $w_j$  encodes a  $C \times C$  confusion matrix  $\mathbf{M}_j$ , where an entry  $\mathbf{M}_j(m, n)$  denotes person  $j$ 's probability of predicting class  $n$  when the true class is  $m$ . Here,  $p(z_{ij}|y_i, w_j) = \mathbf{M}_j(y_i, z_{ij})$ ; the model is assuming  $p(z_{ij}|y_i = c, w_j)$  is a multinomial distribution with parameters  $\mu_j^c = [\mathbf{M}_j(c, 1), \dots, \mathbf{M}_j(c, C)]$  for each value of  $c$ . We will place Dirichlet priors  $\text{Dir}(n_\beta \alpha^c)$  on  $\mu_j^c$ , where  $n_\beta$  is the strength of the prior, and  $\alpha^c$  is estimated by pooling across all workers. We will also place a Dirichlet prior  $\text{Dir}(n_\beta \alpha)$  on  $\alpha^c$ , with  $\alpha$  acting as a global hyper-parameter that provides the likelihood of any worker labeling a class correctly. Because the Dirichlet distribution is the conjugate prior of the multinomial distribution, the computation of each entry  $k$  from  $1 \dots C$  in the skill vector  $\mu_j^c$  for a single worker  $j$  and each class  $c$  is done by counting agreements:

$$\mu_{j,k}^c = \frac{n_\beta \alpha_k^c + \sum_{i \in \mathcal{I}_j} 1[z_{ij} = k, \bar{y}_i = k, |\mathcal{W}_i| > 1] - 1}{n_\beta \alpha_0^c + \sum_{i \in \mathcal{I}_j} 1[\bar{y}_i = k, |\mathcal{W}_i| > 1] - C} \quad (4.5)$$

Where  $\alpha_0^c = \sum_k \alpha_k^c$ . The pooled worker parameters  $\alpha^c$  are estimated in a similar way.

### Taxonomic Models

Multinomial models are useful because they model commonly confused classes, however they have far more parameters than the binomial models. These models quickly become intractable as the total number of classes  $C$  gets large. For example, if there are  $10^4$  classes, we would be attempting to estimate a matrix  $\mathbf{M}_j$  with  $10^8$  entries for each worker  $j$ . This is statistically and computationally intractable. However, when the number of classes gets large, there often exists a taxonomy used to organize them (e.g. the Linnaean taxonomy for biological classification). We can use this taxonomy to reduce the number of parameters in a multinomial model.

**Taxonomic Per Class Multinomial:** We will assume a taxonomy of classes that is  $L$  levels deep and associate a confusion matrix with each node in the taxonomy (e.g., if we know the genus of an observation from iNaturalist, assume each worker has a confusion matrix among species within that genus). For the taxonomic model, let  $y_i^l$  denote the node in the taxonomy at level  $l$  that class  $y_i$  belongs to, such that  $y_i^0$  is the root node and  $y_i^L$  is the leaf node (i.e., species label). Similarly, let  $z_{ij}^l$  denote the node in the taxonomy at level  $l$  that class  $z_{ij}$  belongs to. In this model,  $p(z_{ij}^l | y_i^l, w_j, y_i^{l-1} = z_{ij}^{l-1}) = \mathbf{M}_j^{y_i^{l-1}}(y_i^l, z_{ij}^l)$ , where  $\mathbf{M}_j^{y_i^{l-1}}$  is a confusion matrix associated with node  $y_i^{l-1}$  in the taxonomy; the assumption is that for each value of

$y_i^l, z_{ij}^l$  is multinomial with a vector  $\mathbf{M}_j^{y_i^{l-1}}(y_i^l, \cdot)$  of parameters of size equal to the number of child nodes. The term  $y_i^{l-1} = z_{ij}^{l-1}$  denotes the condition that the parent node classification is known. Suppose, however, that worker  $j$  is wrong about both the species and genus. We must also model  $p(z_{ij}^l | y_i^l, w_j, y_i^{l-1} \neq z_{ij}^{l-1})$ . In our model we assume that worker  $j$  predicts each class  $z_{ij}^l$  with some probability irrespective of the true class (i.e.  $p(z_{ij}^l | y_i^l, w_j, y_i^{l-1} \neq z_{ij}^{l-1}) = \mathbf{N}_j^{z_{ij}^{l-1}}(z_{ij}^l)$  is multinomial with a parameter for each possible child node). The taxonomic model results in the following values that can be plugged into Equation 4.1:

$$p(z_{ij} | y_i, w_j) = \prod_{l=1}^L p(z_{ij}^l | y_i^l, w_j), \quad (4.6)$$

$$p(z_{ij}^l | y_i^l, w_j) = \begin{cases} \mathbf{M}_j^{y_i^{l-1}}(y_i^l, z_{ij}^l) & \text{if } y_i^{l-1} = z_{ij}^{l-1} \\ \mathbf{N}_j^{z_{ij}^{l-1}}(z_{ij}^l) & \text{otherwise} \end{cases} \quad (4.7)$$

Note that in totality, for each node  $n$  in the taxonomy, we have associated a confusion matrix  $\mathbf{M}_j^n$  with a row for each child of  $n$ , and a vector of probabilities  $\mathbf{N}_j^n$  with an entry for each child. If the taxonomy is relatively balanced, this model has far fewer parameters than the flat multinomial model (linear in the number of classes rather than quadratic). To make estimating worker parameters more robust, we will again make use of a tiered system of priors (e.g. Dirichlet priors on all multinomial parameters) that are computed by pooling across all workers at each node. However, if this is still too many parameters, we can fall back to modeling the probability that a person is correct as a binomial distribution with a parameter per child node (i.e. the **taxonomic per class binomial** model), or even just one parameter for all children (i.e. the **taxonomic single binomial** model), assuming other class responses  $z_{ij}^l \neq y_i^l$  have probability proportional to their priors. See Table 4.1 for an overview of all models.

### Taxonomic Predictions

Thus far, we have assumed that a worker always predicts a class of the finest possible granularity (i.e. species level). An alternate UI can allow a worker to predict an internal node in the taxonomy if unsure of the exact class, i.e. applying the “hedging your bets” Deng et al., 2012 method to human classifiers. In Figure 4.1, this would be akin to one of the identifiers specifying the family Ardeidae, which includes both Snowy Egret and Great Egret. Let  $\text{level}(z_{ij})$  be the level of this prediction. Note that  $z_{ij}^l$  is valid only for  $l \leq \text{level}(z_j)$ . The taxonomic model in Section 4.4 works

after an update of Equation 4.6 to  $p(z_{ij}|y_i, w_j) = \prod_{l=1}^{\text{level}(z_{ij})} p(z_{ij}^l|y_i^l, w_j)$ . This works even if different workers provide different levels of taxonomic predictions.

### Dependent Labels

In Section 4.4, we assumed each worker independently guesses the class of image  $i$ . We now turn to the situation described in Figure 4.1: a user submits an observation  $x_i$  and an initial identification  $z_{i,j_i^1}$ , where  $j_i^t$  denotes the  $t$ th worker that labeled image  $i$ . A notification of the observation is sent to users that have subscribed to the taxa  $z_{i,j_i^1}$  or to that particular geographic region (the rest of the community is not explicitly notified but can find the observation when browsing the site). Each subsequent identifier  $j_i^t, t > 1$  can see the details of the observation  $x_i$  and all identifications made by previous users  $H_i^{t-1} = \{z_{i,j_i^1}, z_{i,j_i^2}, \dots, z_{i,j_i^{t-1}}\}$ . Users can assess the experience of a previous identifier  $j$  by viewing all of their observations  $X_j$  and all of their identifications  $Z_j$ . Additionally, users are able to discuss the identifications through comments.

In this setting, we can adapt Equation 4.1 to

$$\begin{aligned} p(y_i|Z_i) &= p(y_i|H_i^{|\mathcal{W}_i|}) \\ &= \frac{p(y_i) \prod_{t=1}^{|\mathcal{W}_i|} p(z_{i,j_i^t}|y_i, H_i^{t-1}, w_{j_i^t})}{\sum_{y=1}^C p(y) \prod_{t=1}^{|\mathcal{W}_i|} p(z_{i,j_i^t}|y, H_i^{t-1}, w_{j_i^t})} \end{aligned} \quad (4.8)$$

There are many possible choices for modeling  $p(z_{i,j_i^t}|y_i, H_i^{t-1}, w_{j_i^t})$ . The simplest option assumes each worker ignores all prior responses; i.e.,  $p(z_{i,j_i^t}|y_i, H_i^{t-1}, w_{j_i^t}) = p(z_{i,j_i^t}|y_i, w_{j_i^t})$ . In practice, however, worker  $j_i^t$ 's response will probably be biased toward agreeing with prior responses  $H_i^{t-1}$ , making a prediction combining both evidence from analyzing prior responses and from observing the image itself. The weight of this evidence should increase with the number of prior responses and could vary based on worker  $j_i^t$ 's assessment of other worker's skill levels. In our model, we assume that worker  $j_i^t$  weights each possible response  $z_{i,j_i^t}$  (worker  $j_i^t$ 's perception of the class of image  $i$ ) with a term  $p_{j_i^t}(H_i^{t-1}|z_{i,j_i^t})$  (worker  $j_i^t$ 's perception of the probability of prior responses given that class).  $p(z_{i,j_i^t}|y_i, H_i^{t-1}, w_{j_i^t})$  can then be expressed as:

$$\begin{aligned} p(z_{i,j_i^t}|y_i, H_i^{t-1}, w_{j_i^t}) &= \frac{p(z_{i,j_i^t}, H_i^{t-1}|y_i, w_{j_i^t})}{p(H_i^{t-1}|y_i, w_{j_i^t})} \\ &= \frac{p(z_{i,j_i^t}|y_i, w_{j_i^t})p_{j_i^t}(H_i^{t-1}|z_{i,j_i^t}, w_{j_i^t})}{\sum_z p(z|y_i, w_{j_i^t})p_{j_i^t}(H_i^{t-1}|z, w_{j_i^t})} \end{aligned} \quad (4.9)$$



where  $p(z_{i,j_i^t} | y_i, w_{j_i^t})$  is modeled using a method described in Section 4.4. Worker  $j_i^t$  might choose to treat each prior response as independent sources of information  $p_{j_i^t}(H_i^{t-1} | z_{i,j_i^t}, w_{j_i^t}^j) = \prod_{s=1}^{t-1} p_{j_i^t}(z_{i,j_i^s} | z_{i,j_i^t}, w_{j_i^s}^j)$  where we have used the notation  $w_k^j$  to denote parameters for worker  $j$ 's perception of worker  $k$ 's skill. Alternatively, worker  $j$  may choose to account for the fact that earlier responses were also biased by prior responses using similar assumptions to those we made in Equation 4.9, resulting in a recursive definition/computation of  $p_{j_i^t}(H_i^{t-1} | z_{i,j_i^t}, w_{j_i^t}^j) =$

$$\begin{cases} \frac{p_{j_i^t}(z_{i,j_i^{t-1}} | z_{i,j_i^t}, w_{j_i^{t-1}}^j) p_{j_i^{t-1}}(H_i^{t-2} | z_{i,j_i^{t-1}}, w_{j_i^{t-2}}^j)}{\sum_z p_{j_i^t}(z | z_{i,j_i^t}, w_{j_i^{t-1}}^j) p_{j_i^{t-1}}(H_i^{t-2} | z, w_{j_i^{t-2}}^j)} & \text{if } t > 1 \\ p_{j_i^t}(z_{i,j_i^{t-1}} | z_{i,j_i^t}, w_{j_i^{t-1}}^j) & \text{if } t = 1 \end{cases} \quad (4.10)$$

The last choice to make is how to model probabilities of the form  $p_j(z_k | z_j, w_k^j)$  (i.e. worker  $j$ 's perception of worker  $k$ 's responses). One model that keeps the number of parameters low is a binomial distribution: worker  $j$  assumes other workers are correct with probability  $\rho_j$ ; when they are incorrect, they respond proportionally to class priors:

$$p_j(z_k | z_j, w_k^j) = \begin{cases} \rho_j & \text{if } z_k = z_j \\ (1 - \rho_j)p(z_j) & \text{otherwise} \end{cases} \quad (4.11)$$

Here,  $\rho_j$  is a learned parameter expressing worker  $j$ 's trust in the responses of other workers.

#### 4.5 Taking Pixels into Account

Rather than relying on class priors  $p(y_i)$ , we can make use of a computer vision model with parameters  $\theta$  that can predict the probability of each class occurring in each image  $x_i \in X$ . This results in an update to Equation 4.1, changing  $p(y_i)$  to  $p(y_i | x_i, \theta)$ . We use a computer vision model similar to the general purpose binary computer vision system trained by Branson et al. (Branson, Van Horn, and Perona, 2017). We extract ‘‘PreLogit’’ features  $\phi(x_i)$  from an Inception-v3 (Szegedy et al., 2016) CNN for each image  $i$ , and use these features (fixed for all iterations) to train the weights  $\theta$  of a linear SVM (using a one-vs-rest strategy), followed by probability calibration using Platt scaling (Platt et al., 1999). We use stratified cross-validation to construct training and validation splits that contain at least one sample from each class. This results in probability estimates  $p(y_i | x_i, \theta) = \sigma(\gamma \theta \cdot \phi(x_i))$ , where  $\gamma$  is the probability calibration scalar from Platt scaling, and  $\sigma(\cdot)$  is the sigmoid function. Fine-tuning a CNN on each iteration would lead to better performance (Agrawal,

Method	Label Error Rate (%)
(Ghosh, Kale, and McAfee, 2011), (Dalvi et al., 2013)	27.78
Majority Vote	24.07
<b>Flat Multinomial</b> , (Dawid and Skene, 1979), (Welinder et al., 2010), (Karger, Oh, and D. Shah, 2013)	11.11
<b>Flat Multinomial-CV</b> , (Tian and Zhu, 2015), (Y. Zhang et al., 2014)*	10.19

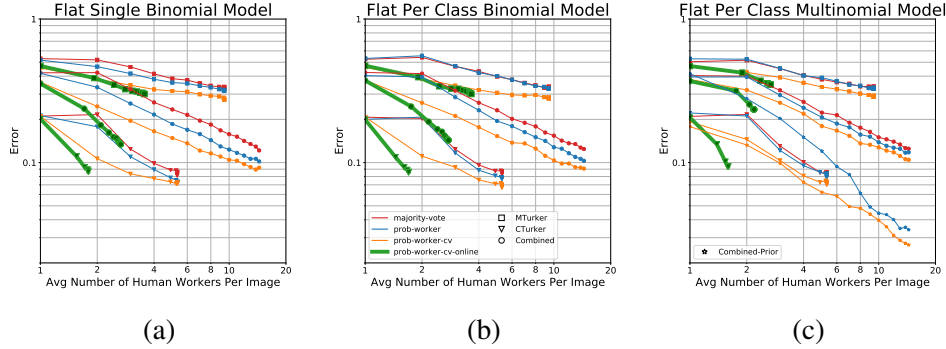
Table 4.2: Label error rates of different worker skill models on the binary Bluebird dataset (Welinder et al., 2010) after receiving *all* 4,212 annotations. Our methods (**Flat Multinomial**, and **Flat Multinomial-CV**) are competitive with other methods. \*(Y. Zhang et al., 2014) mistakenly reported 10.09.

Girshick, and Malik, 2014; Oquab et al., 2014; Yosinski et al., 2014) but is out of scope.

## 4.6 Experiments

We evaluate the proposed models on data collected from paid workers through Amazon Mechanical Turk (MTurk) and from non-paid citizen scientists who are members of the Cornell Lab of Ornithology (Lab of O) or iNaturalist (iNat). We follow a similar evaluation protocol to (Branson, Van Horn, and Perona, 2017) and use Algorithm 1 from that work to run the experiments. For models that assume worker labels are independent, we simulate multiple trials by adding worker labels in random order. For lesion studies, we simply turn off parts of the model by preventing those parts from updating. The tag *prob-worker* means that a global prior is computed across all workers, and per worker skill model was used; the tag *online* means that online crowdsourcing was used (with risk threshold parameter  $\tau_\epsilon = .02$ ), and the tag *cv* means that computer vision probabilities were used instead of class priors. **Bluebirds** To gauge the effectiveness of our model against prior work, we run our models on the *binary* bluebird dataset from (Welinder et al., 2010). This dataset has a total of 108 images and 39 MTurkers labeled every image for a total of 4,212 annotations. Table 4.2 has the final label error rates of different worker skill models when *all* annotations are made available. Our offline, flat multinomial models are competitive with other offline methods.

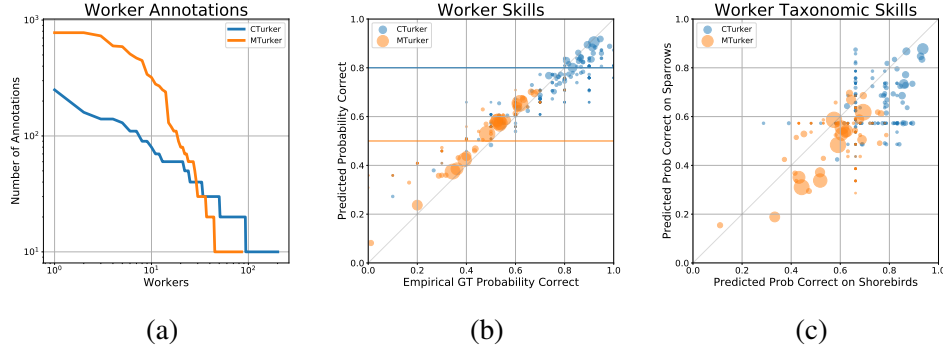
**NABirds** This experiment was designed to test our models in a traditional dataset collection situation where labeling tasks are posted to a crowdsourcing website and



**Figure 4.2: Crowdsourcing Multiclass Labels with MTurkers and CTurkers:** These figures show results from our flat models on a dataset of 69 species of birds with labels from Amazon Mechanical Turk workers (MTurkers) and citizen scientists (CTurkers). Each model was run on a dataset that consisted of: just MTurkers (squares), just CTurkers (triangles) or a combination of the two (circles). When our full framework is used (prob-worker-cv-online, green lines) we can achieve the same error as majority vote (red lines) with much fewer labels per image. When we use our framework in an offline setting (prob-worker-cv and prob-worker, orange and blue curves), we can achieve a lower error than majority vote with the same number of labels. When initialized with generic priors, the single binomial model achieves the lowest error, followed by the per class binomial and the multinomial model. However, if domain knowledge is used to initialize the global priors to more reasonable values, the multinomial model can achieve impressively low error (the star lines in (c)).

responses are collected independently. We constructed a labeling interface that showed workers a sequence of 10 images and asked them to classify each image into one of 69 different bird species by using an auto complete box or by browsing a gallery of representative photos for each species. We used 998 images, all sampled from either shorebird or sparrow species, from the NABirds dataset (Van Horn et al., 2015). We collected responses from both MTurkers and citizen scientists from the Lab of O (CTurkers). Figure 4.3a shows the contribution of annotations from the workers. We had a total of 86 MTurkers provide 9,391 labels and a total of 202 CTurkers provide 5,300 labels. For these experiments, we made the gallery of example images (3 to 5 images per species) available to the computer vision system during training. This ensured that we could construct at least 3 cross validation splits when calibrating the computer vision probabilities in the early stages of the algorithm.

All models were initialized with uniform class priors, a probability of 0.5 that an MTurker will label a class correctly, and a probability of 0.8 that a CTurker will label



**Figure 4.3: MTurker and CTurker Worker Analysis:** Figure (a) shows the contribution of labels per worker from MTurkers and CTurkers. On average we have less than one label from each worker for each of the 69 classes, emphasizing the need to pool data across workers for use as priors. Figure (b) shows the predicted probability of a worker providing a correct label  $m_j$  plotted against the empirical ground truth probability for the single binomial prob-worker-cv model from 4.2a. The size of each dot is proportional to the number of annotations that worker contributed to the dataset. Solid lines mark the priors. We can see that the model’s predictions correlate well with the empirical ground truths. Figure (c) shows the predicted worker skill for correctly labeling the species of a sparrow vs. correctly labeling the species of a shorebird. These skill estimates came from a taxonomic binomial model with one subtree corresponding to sparrows and the other corresponding to shorebirds. In real applications, we can use these skill estimates to direct images to proficient labelers.

a class correctly. This means the global Dirichlet priors (used in the multinomial models) had a value of 0.8 at the true class index and 0.003 otherwise for the CTurkers. These are highly conservative priors. For each of our three flat models we conducted three experiments: using MTurk data only, using CTurk data only, and using both MTurk and CTurk data together (“Combined” in the plots). Figure 4.2 shows the results. First, we note that when a computer vision system is utilized in an online fashion (prob-worker-cv-online), we see a significant decrease in the average number of labels per image to reach the same performance as majority vote using all of the data (e.g. a  $5.4\times$  decrease in the single binomial combined setting). In the offline setting (prob-worker-cv), the computer vision models decrease the final error compared to majority vote (e.g. 25% decrease in error in the single binomial combined setting). When considering our probabilistic model without computer vision (prob-worker) the single binomial model consistently achieved the lowest error, followed by the binomial per class model and then the multinomial model. This is not unexpected, as we anticipated the larger capacity models to struggle with

the sparseness of data (i.e. on average we had 0.75 labels per class per worker in the combined setting). However, the fact that they approach similar performance to the single binomial model highlights the usefulness of our tiered prior system and the ability to pool data across all of the workers. Our global prior initializations are purposefully on the conservative side, however in a real application setting, a user of this framework can initialize the priors using domain knowledge or a small amount of ground truth data. Figure 4.2c shows the dramatic effect of using more informative priors in the combined setting (prob-worker-cv and prob-worker in the Combined-Prior setting). These models were initialized with priors that were computed on a small held out set of worker annotations with ground truth labels and achieved the lowest error (0.03, for prob-worker-cv, a 79% decrease from majority vote) on the dataset.

Figure 4.3b shows the predicted  $m_j$  values learned by the single binomial model plotted against the empirical ground truth in the combined setting. We can see that the model’s predictions correlate well with the empirical estimates, with increasing precision as the number of annotations increases (size of the dots). To further investigate the worker skills, we constructed a simple 2 level taxonomy and placed the shorebirds and sparrows in their own flat subtrees. By running our taxonomic binomial model, we are able to learn a skill for each group separately, rendered in Figure 4.3c. We can see that both MTurkers and CTurkers have a higher probability of predicting shorebirds correctly than sparrows. In real applications, we can use these skill estimates to direct images to proficient labelers.

**iNaturalist** This experiment was designed to test our models in a classification situation that mimics the real world scenario of websites like iNat, see Figure 4.1. We obtained a database export from iNat and cleaned the data using the following three steps: (1) we select observations and identifications from a subset of the taxonomy (e.g. species of birds); (2) for each observation, we keep only the first identification from each user (i.e. we do not allow users to change their minds); and (3) to facilitate experiments, we keep all observations that have a ground truth label at the species level (i.e. leaf nodes of the taxonomy). For the experiments presented below, after performing the previous steps, we selected a subset of 30 species of birds and 1000 observations from each species to analyze. In this 30k image subset we have 5,643 workers that provided a total of 98,849 labels; Figure 4.4c shows the distribution of worker annotations. The taxonomy associated with these 30 species consisted of 44 nodes with a max depth of 3. For these experiments we did not

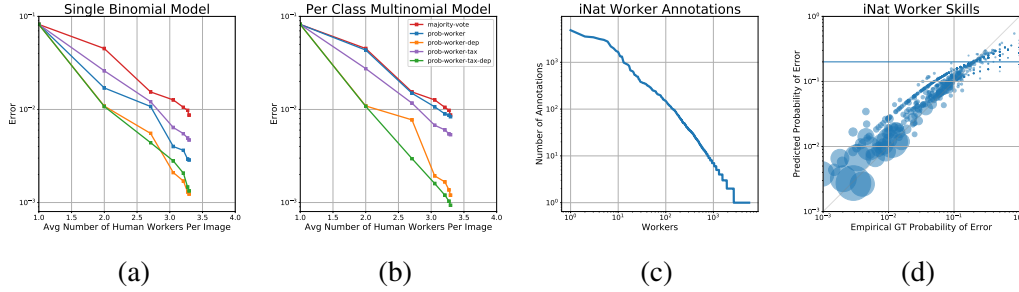


Figure 4.4: **iNaturalist Birds** Figures (a) and (b) show the errors achieved on a dataset of 30 bird species from iNaturalist for the single binomial and multinomial models respectively. Each model was evaluated in several configurations: “prob-worker” assumes a flat list of species. “prob-worker-tax” takes advantage of a taxonomy across the species, allowing workers to provide non-leaf node annotations and reducing the number of parameters in the multinomial model from 900 to 167. “prob-worker-dep” assumes a flat list of species, but models the dependence between the worker labels. “prob-worker-tax-dep” uses a taxonomy across the species and models the dependence between worker labels. All models did at least as well as majority vote, with dependence modeling providing a significant decrease in error. The lowest error was achieved by the multinomial prob-worker-tax-dep model that was capable of modeling species confusions and label dependencies, decreasing error by 90% compared to majority vote. Figure (c) shows the distribution of labels per worker, emphasizing a long tail of worker contributions. Figure (d) shows the predicted probability of error ( $1 - m_j$ ) for each worker plotted against the empirical ground truth probability of error for the single binomial prob-worker-dep model, with the radius of a dot proportional to the number of annotations contributed by that worker. The solid blue line is the global prior value. More active identifiers are less likely to make errors, and our model skill estimates correlate well with the empirical ground truths.

utilize a computer vision system. Class priors were initialized to be uniform, skill priors were initialized assuming that iNat users are 80% correct. Worker labels are added to the images sequentially by their time stamp, so only a single pass through the data is possible.

Figures 4.4a and 4.4b show the results for our single binomial and multinomial models respectively. For each model we used flat and taxonomic (-tax) versions, and we turned on (-dep) and off label dependence modeling, for a total of 4 variations of each model. We can see that all of our models are at least as good as majority vote. Adding dependence modeling to the flat models provides a significant decrease in error: a 59% decrease for the flat single binomial model, and an 85% decrease for the flat multinomial model. The taxonomic single binomial model (with 14

parameters per worker) did slightly worse than the flat single binomial model (with 1 parameter per worker). However, the taxonomic multinomial model (with 167 parameters per worker) decreased error by 36% compared to the flat multinomial model (with 900 parameters per worker). Finally, adding dependence modeling to the taxonomic models provided a further decrease in error, with the taxonomic multinomial model performing the best and decreasing error by 90% over majority vote, corresponding to 28 total errors. While a majority of those errors were true mistakes, an inspection of a few revealed errors in the ground truth labels of the iNat dataset. Figure 4.1 is actually an example of one of those mistakes. Further, the observation (<https://tinyurl.com/ycu92cas>) associated with the second “riskiest” image (using the computed Bayes risk of the predicted label  $\mathcal{R}(\bar{y}_i)$ ) turned out to be another mistake, advocating the use of these models as a way of sorting the observations for expert review. Figure 4.4d shows the predicted probability of a worker labeling incorrectly ( $1 - m_j$ ) for the flat single binomial model with dependence modeling from Figure 4.4a. We can see that the model’s skill predictions correlate well with the empirical ground truth skills.

## 4.7 Conclusion

We introduced new multiclass annotation models that can be used in the online crowdsourcing framework of Branson et al. (Branson, Van Horn, and Perona, 2017). We explored several variants of a worker skill model using a variety of parameterizations and we showed how to harness a taxonomy to reduce the number of parameters when the number of classes is large. As an additional benefit, our taxonomic models are capable of processing worker labels from anywhere in the taxonomy rather than just leaf nodes. Finally, we presented techniques for modeling the dependence of worker labels in tasks where workers can see a prior history of identifications. Our models consistently outperform majority vote, either reaching a similar error with far fewer annotations or achieving a lower error with the same number of annotations. Future work involves modeling “schools of thought” among workers and using their skill estimates to explore human teaching.

## Acknowledgments

This work was supported by a Google Focused Research Award. We thank Oisín Mac Aodha for useful discussions.

## References

- Agrawal, Pulkit, Ross Girshick, and Jitendra Malik (2014). “Analyzing the performance of multilayer neural networks for object recognition”. In: *European Conference on Computer Vision*. Springer, pp. 329–344.
- Branson, Steve, Grant Van Horn, and Pietro Perona (2017). “Lean Crowdsourcing: Combining Humans and Machines in an Online System”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7474–7483. DOI: 10.1109/CVPR.2017.647.
- Chen, Guangyong et al. (2017). “Learning to Aggregate Ordinal Labels by Maximizing Separating Width”. In: *International Conference on Machine Learning*, pp. 787–796.
- Dalvi, Nilesh et al. (2013). “Aggregating crowdsourced binary ratings”. In: *Proceedings of the 22nd international conference on World Wide Web*. ACM, pp. 285–294.
- Dawid, Alexander Philip and Allan M Skene (1979). “Maximum likelihood estimation of observer error-rates using the EM algorithm”. In: *Applied statistics*, pp. 20–28.
- Deng, Jia et al. (2012). “Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 3450–3457.
- Ghosh, Arpita, Satyen Kale, and Preston McAfee (2011). “Who moderates the moderators?: crowdsourcing abuse detection in user-generated content”. In: *Proceedings of the 12th ACM conference on Electronic commerce*. ACM, pp. 167–176.
- He, Kaiming et al. (2015). “Deep residual learning for image recognition”. In: *arXiv preprint arXiv:1512.03385*.
- Huang, Gao et al. (2017). “Densely connected convolutional networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- Jin, Rong and Zoubin Ghahramani (2002). “Learning with multiple labels”. In: *Advances in neural information processing systems*, pp. 897–904.
- Kamar, Ece, Severin Hacker, and Eric Horvitz (2012). “Combining human and machine intelligence in large-scale crowdsourcing”. In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, pp. 467–474.
- Karger, David R, Sewoong Oh, and Devavrat Shah (2011). “Iterative learning for reliable crowdsourcing systems”. In: *Advances in neural information processing systems*, pp. 1953–1961.



- Karger, David R, Sewoong Oh, and Devavrat Shah (2013). “Efficient crowdsourcing for multi-class labeling”. In: *ACM SIGMETRICS Performance Evaluation Review* 41.1, pp. 81–92.
- (2014). “Budget-optimal task allocation for reliable crowdsourcing systems”. In: *Operations Research* 62.1, pp. 1–24.
- Kovashka, A. et al. (2016). “Crowdsourcing in Computer Vision”. In: *ArXiv e-prints*. arXiv: 1611.02145 [cs.CV]. URL: <https://arxiv.org/abs/1611.02145>.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks.” In: *NIPS*.
- Li, Hongwei, Bin Yu, and Dengyong Zhou (2013). “Error rate analysis of labeling by crowdsourcing”. In: *ICML Workshop: Machine Learning Meets Crowdsourcing. Atalanta, Georgia, USA*.
- Little, Greg et al. (2010). “Exploring iterative and parallel human computation processes”. In: *Proceedings of the ACM SIGKDD workshop on human computation*. ACM, pp. 68–76.
- Littlestone, Nick and Manfred K Warmuth (1994). “The weighted majority algorithm”. In: *Information and computation* 108.2, pp. 212–261.
- Liu, Qiang, Jian Peng, and Alexander T Ihler (2012). “Variational inference for crowdsourcing”. In: *Advances in Neural Information Processing Systems*, pp. 692–700.
- Long, Chengjiang, Gang Hua, and Ashish Kapoor (2013). “Active visual recognition with expertise estimation in crowdsourcing”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3000–3007.
- Mora, Camilo et al. (2011). “How many species are there on Earth and in the ocean?” In: *PLoS biology* 9.8, e1001127.
- Ok, Jungseul et al. (2016). “Optimality of Belief Propagation for Crowdsourced Classification”. In: *arXiv preprint arXiv:1602.03619*.
- Oquab, Maxime et al. (2014). “Learning and transferring mid-level image representations using convolutional neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724.
- Platt, John et al. (1999). “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods”. In: *Advances in large margin classifiers* 10.3, pp. 61–74.
- Raykar, Vikas C et al. (2010). “Learning from crowds”. In: *Journal of Machine Learning Research* 11.Apr, pp. 1297–1322.
- Shah, Nihar Bhadresh and Denny Zhou (2015). “Double or nothing: Multiplicative incentive mechanisms for crowdsourcing”. In: *Advances in Neural Information Processing Systems*, pp. 1–9.

- Smyth, Padhraic et al. (1995). “Inferring ground truth from subjective labelling of venus images”. In:
- Sullivan, Brian L et al. (2014). “The eBird enterprise: an integrated approach to development and application of citizen science”. In: *Biological Conservation* 169, pp. 31–40.
- Szegedy, Christian et al. (2016). “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.
- Tang, Wei and Matthew Lease (2011). “Semi-supervised consensus labeling for crowdsourcing”. In: *SIGIR 2011 workshop on crowdsourcing for information retrieval (CIR)*, pp. 1–6.
- Tian, Tian and Jun Zhu (2015). “Max-margin majority voting for learning from crowds”. In: *Advances in Neural Information Processing Systems*, pp. 1621–1629.
- Ueda, K (2017). “iNaturalist Research-grade Observations via GBIF.org.” In: URL: <https://doi.org/10.15468/ab3s5x>.
- Van Horn, Grant et al. (2015). “Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604. doi: 10.1109/CVPR.2015.7298658.
- Vempaty, Aditya, Lav R Varshney, and Pramod K Varshney (2014). “Reliable crowdsourcing for multi-class labeling using coding theory”. In: *IEEE Journal of Selected Topics in Signal Processing* 8.4, pp. 667–679.
- Welinder, Peter et al. (2010). “The multidimensional wisdom of crowds”. In: *Advances in neural information processing systems*, pp. 2424–2432.
- Whitehill, Jacob et al. (2009). “Whose vote should count more: Optimal integration of labels from labelers of unknown expertise”. In: *Advances in neural information processing systems*, pp. 2035–2043.
- Yosinski, Jason et al. (2014). “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems*, pp. 3320–3328.
- Zhang, Jing et al. (2016). “Multi-class ground truth inference in crowdsourcing with clustering”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.4, pp. 1080–1085.
- Zhang, Yuchen et al. (2014). “Spectral methods meet EM: A provably optimal algorithm for crowdsourcing”. In: *Advances in neural information processing systems*, pp. 1260–1268.
- Zhou, Dengyong et al. (2014). “Aggregating Ordinal Labels from Crowds by Minimax Conditional Entropy.” In: *ICML*, pp. 262–270.

Zhou, Denny et al. (2012). “Learning from the wisdom of crowds by minimax entropy”. In: *Advances in Neural Information Processing Systems*, pp. 2195–2203.

## *Chapter 5*

### BUILDING A BIRD RECOGNITION APP AND LARGE SCALE DATASET WITH CITIZEN SCIENTISTS: THE FINE PRINT IN FINE-GRAINED DATASET COLLECTION

Van Horn, Grant et al. (2015). “Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604. DOI: 10.1109/CVPR.2015.7298658.

#### **5.1 Abstract**

We introduce tools and methodologies to collect high quality, large scale, fine-grained, computer vision datasets using citizen scientists – crowd annotators who are passionate and knowledgeable about specific domains such as birds or airplanes. We worked with citizen scientists and domain experts to collect NABirds, a new high quality dataset containing 48,562 images of North American birds with 555 categories, part annotations and bounding boxes. We find that citizen scientists are significantly more accurate than Mechanical Turkers at zero cost. We worked with bird experts to measure the quality of popular datasets like CUB-200-2011 and ImageNet and found class label error rates of at least 4%. Nevertheless, we found that learning algorithms are surprisingly robust to annotation errors, and this level of training data corruption can lead to an acceptably small increase in test error if the training set has sufficient size. At the same time, we found that an expert-curated high quality test set like NABirds is necessary to accurately measure the performance of fine-grained computer vision systems. We used NABirds to train a publicly available bird recognition service deployed on the web site of the Cornell Lab of Ornithology.<sup>1</sup>

#### **5.2 Introduction**

Computer vision systems – catalyzed by the availability of new, larger scale datasets like ImageNet (Jia Deng, Dong, et al., 2009) – have recently obtained remarkable performance at object recognition (Krizhevsky, Sutskever, and Hinton, 2012; Taigman et al., 2014) and detection (Girshick et al., 2013). Computer vision has entered

---

<sup>1</sup>[merlin.allaboutbirds.org](http://merlin.allaboutbirds.org)

an era of big data, where the ability to collect larger datasets – larger in terms of the number of classes, the number of images per class, and the level of annotation per image – appears to be paramount for continuing performance improvement and expanding the set of solvable applications.

Unfortunately, expanding datasets in this fashion introduces new challenges beyond just increasing the amount of human labor required. As we increase the number of classes of interest, classes become more fine-grained and difficult to distinguish for the average person (and the average annotator), more ambiguous, and less likely to obey an assumption of mutual exclusion. The annotation process becomes more challenging, requiring an increasing amount of skill and knowledge. Dataset *quality* appears to be at direct odds with dataset *size*.

In this chapter, we introduce tools and methodologies for constructing large, high quality computer vision datasets, based on tapping into an alternate pool of crowd annotators – citizen scientists. Citizen scientists are nonprofessional scientists or enthusiasts in a particular domain such as birds, insects, plants, airplanes, shoes, or architecture. Citizen scientists contribute annotations with the understanding that their expertise and passion in a domain of interest can help build tools that will be of service to a community of peers. Unlike workers on Mechanical Turk, citizen scientists are unpaid. Despite this, they produce higher quality annotations due to their greater expertise and the absence of spammers. Additionally, citizen scientists can help define and organically grow the set of classes and its taxonomic structure to match the interests of real users in a domain of interest. Whereas datasets like ImageNet (J. Deng et al., 2009) and CUB-200-2011 (Wah et al., 2011) have been valuable in fostering the development of computer vision algorithms, the particular set of categories chosen is somewhat arbitrary and of limited use to real applications.

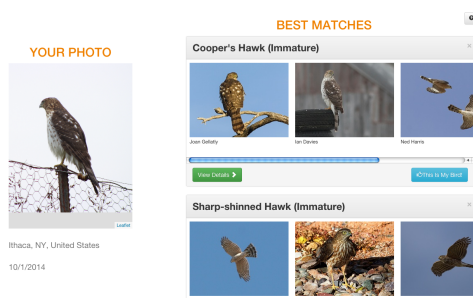


Figure 5.1: **Merlin Photo ID**: a publicly available tool for bird species classification built with the help of citizen scientists. The user uploaded a picture of a bird, and server-side computer vision algorithms identified it as an immature Cooper’s Hawk.

The drawback of using citizen scientists instead of Mechanical Turkers is that the throughput of collecting annotations may be lower, and computer vision researchers must take the time to figure out how to partner with different communities for each domain.

We collected a large dataset of 48,562 images over 555 categories of birds with part annotations and bounding boxes for each image, using a combination of citizen scientists, experts, and Mechanical Turkers. We used this dataset to build a publicly available application for bird species classification. In this chapter, we provide details and analysis of our experiences with the hope that they will be useful and informative for other researchers in computer vision working on collecting larger fine-grained image datasets. We address questions like: what is the relative skill level of different types of annotators (MTurkers, citizen scientists, and experts) for different types of annotations (fine-grained categories and parts)? What are the resulting implications in terms of annotation quality, annotation cost, human annotator time, and the time it takes a requester to finish a dataset? Which types of annotations are suitable for different pools of annotators? What types of annotation GUIs are best for each respective pool of annotators? How important is annotation quality for the accuracy of learned computer vision algorithms? How significant are the quality issues in existing datasets like CUB-200-2011 and ImageNet, and what impact has that had on computer vision performance?

We summarize our contributions below:

1. Methodologies to collect high quality, fine-grained computer vision datasets using a new type of crowd annotators: citizen scientists.
2. NABirds: a large, high quality dataset of 555 categories curated by experts.
3. Merlin Photo ID: a publicly available tool for bird species classification.
4. Detailed analysis of annotation quality, time, cost, and throughput of MTurkers, citizen scientists, and experts for fine-grained category and part annotations.
5. Analysis of the annotation quality of the popular datasets CUB-200 and ImageNet.
6. Empirical analysis of the effect that annotation quality has when training state-of-the-art computer vision algorithms for categorization.

A high-level summary of our findings is: (a) citizen scientists have 2-4 times lower error rates than MTurkers at fine-grained bird annotation, while annotating images faster and at zero cost. Over 500 citizen scientists annotated images in our dataset – if we can expand beyond the domain of birds, the pool of possible citizen scientist annotators is massive. (b) A curation-based interface for visualizing and manipulating the full dataset can further improve the speed and accuracy of citizen scientists and experts. (c) Even when averaging answers from 10 MTurkers together, MTurkers have a more than 30% error-rate at 37-way bird classification. (d) The general high quality of Flickr search results (84% accurate when searching for a particular species) greatly mitigates the errors of MTurkers when collecting fine-grained datasets. (e) MTurkers are as accurate and fast as citizen scientists at collecting part location annotations. (f) MTurkers have faster throughput in collecting annotations than citizen scientists; however, using citizen scientists it is still realistic to annotate a dataset of around 100k images in a domain like birds in around 1 week. (g) At least 4% of images in CUB-200-2011 and ImageNet have incorrect class labels, and numerous other issues including inconsistencies in the taxonomic structure, biases in terms of which images were selected, and the presence of duplicate images. (h) Despite these problems, these datasets are still effective for computer vision research; when training CNN-based computer vision algorithms with corrupted labels, the resulting increase in test error is surprisingly low and significantly less than the level of corruption. (i) A consequence of findings (a), (d), and (h) is that training computer vision algorithms on unfiltered Flickr search results (with no annotation) can often outperform algorithms trained when filtering by MTurker majority vote.

### 5.3 Related Work

#### **Crowdsourcing with Mechanical Turk**

Amazon’s Mechanical Turk (AMT) has been an invaluable tool that has allowed researchers to collect datasets of significantly larger size and scope than previously possible (Sorokin and Forsyth, 2008; J. Deng et al., 2009; Lin et al., 2014). AMT makes it easy to outsource simple annotation tasks to a large pool of workers. Although these workers will usually be non-experts, for many tasks it has been shown that repeated labeling of examples by multiple non-expert workers can produce high quality labels (Sheng, Provost, and Ipeirotis, 2008; Welinder and Pietro Perona, 2010; Ipeirotis, Provost, et al., 2013). Annotation of fine-grained categories is a possible counter-example, where the average annotator may have little to no prior

knowledge to make a reasonable guess at fine-grained labels. For example, the average worker has little to no prior knowledge as to what type of bird a "Semipalmated Plover" is, and her ability to provide a useful guess is largely dependent on the efforts of the dataset collector to provide useful instructions or illustrative examples. Since our objective is to collect datasets of thousands of classes, generating high quality instructions for each category is difficult or infeasible.

### **Crowdsourcing with expertise estimation**

A possible solution is to try to automatically identify the subset of workers who have adequate expertise for fine-grained classification (Welinder, Branson, et al., 2010; Whitehill et al., 2009; Raykar et al., 2009; Long, Hua, and Kapoor, 2013). Although such models are promising, it seems likely that the subset of Mechanical Turkers with expertise in a particular fine-grained domain is small enough to make such methods impractical or challenging.

### **Games with a purpose**

Games with a purpose target alternate crowds of workers that are incentivized by construction of annotation tasks that also provide some entertainment value. Notable examples include the ESP Game (Von Ahn, 2006), reCAPTCHA (Von Ahn et al., 2008), and BubbleBank (Jia Deng, Krause, and Fei-Fei, 2013). A partial inspiration to our work was Quizz (Ipeirotis and Gabrilovich, 2014), a system to tap into new, larger pools of unpaid annotators using Google AdWords to help find and recruit workers with the applicable expertise.<sup>2</sup> A limitation of games with a purpose is that they require some artistry to design tools that can engage users.

### **Citizen science**

The success of Wikipedia is another major inspiration to our work, where citizen scientists have collaborated to generate a large, high quality web-based encyclopedia. Studies have shown that citizen scientists are incentivized by altruism, sense of community, and reciprocity (Kuznetsov, 2006; Nov, 2007; Yang and Lai, 2010), and such incentives can lead to higher quality work than monetary incentives (Gneezy and Rustichini, 2000).

---

<sup>2</sup>The viability of this approach remains to be seen, as our attempt to test it was foiled by a misunderstanding with the AdWords team.



## Datasets

Progress in object recognition has been accelerated by dataset construction. These advances are fueled both by the release and availability of each dataset but also by subsequent competitions on them. Key datasets/competitions in object recognition include Caltech-101 (Fei-Fei, Fergus, and Pietro Perona, 2006), Caltech-256 (Griffin, Holub, and P Perona, 2007), Pascal VOC (Everingham et al., 2010), and ImageNet/ILSVRC (J. Deng et al., 2009; Russakovsky et al., 2014).

Fine-grained object recognition is no exception to this trend. Various domains have already had datasets introduced including Birds (the CUB-200 (Wah et al., 2011) and recently announced Birdsnap (T. Berg et al., 2014) datasets), Flowers (Nilsback and Zisserman, 2008; Angelova and Zhu, 2013), Dogs and Cats (Khosla et al., 2011; Parkhi et al., 2011; Liu et al., 2012), Stoneflies (Martinez-Munoz et al., 2009), Butterflies (Lazebnik, Schmid, and Ponce, 2004), and Fish (Boom et al., 2014) along with man-made domains such as Airplanes (Maji et al., 2013), Cars (Krause et al., 2013), and Shoes (T. L. Berg, A. C. Berg, and Shih, 2010).

### 5.4 Crowdsourcing with Citizen Scientists

The communities of enthusiasts for a taxon are an untapped work force and partner for vision researchers. The individuals comprising these communities tend to be very knowledgeable about the taxon. Even those that are novices make up for their lack of knowledge with passion and dedication. These characteristics make these communities a fundamentally different work force than the typical paid crowd workers. When building a large, fine-grained dataset, they can be utilized to curate images with a level of accuracy that would be extremely costly with paid crowd workers, see Section 5.6. There is a mutual benefit as the taxon communities gain from having a direct influence on the construction of the dataset. They know their taxon, and their community, better than vision researchers, and so they can ensure that the resulting datasets are directed towards solving real world problems.

A connection must be established with these communities before they can be utilized. We worked with ornithologists at the Cornell Lab of Ornithology to build NABirds. The Lab of Ornithology provided a perfect conduit to tap into the large citizen scientist community surrounding birds. Our partners at the Lab of Ornithology described that the birding community, and perhaps many other taxon communities, can be segmented into several different groups, each with their own particular benefits. We built custom tools to take advantage of each of the segments.

## Experts

Experts are the professionals of the community, and our partners at the Lab of Ornithology served this role. Figure 5.4 is an example of an expert management tool (Vibe<sup>3</sup>) and was designed to let expert users quickly and efficiently curate images and manipulate the taxonomy of a large dataset. Beyond simple image storage, tagging, and sharing, the benefit of this tool is that it lets the experts define the dataset taxonomy as they see fit, and allows for the dynamic changing of the taxonomy as the need arises. For NABirds, an interesting result of this flexibility is that bird species were further subdivided into “visual categories.” A “visual category” marks a sex or age or plumage attribute of the species that results in a visually distinctive difference from other members within the same species, see Figure 5.2. This type of knowledge of visual variances at the species level would have been difficult to capture without the help of someone knowledgeable about the domain.

## Citizen Scientist Experts

After the experts, these individuals of the community are the top tier, most skilled members. They have the confidence and experience to identify easily confused classes of the taxonomy. For the birding community, these individuals were identified by their participation in eBird, a resource that allows birders to record and analyze their bird sightings.<sup>4</sup> Figure 5.3a shows a tool that allows these members to take bird quizzes. The tool presents the user with a series of images and requests the species labels. The user can supply the label using the autocomplete box, or, if they are not sure, they can browse through a gallery of possible answers. At the end of the quiz, their answers can be compared with other expert answers.

---

<sup>3</sup>vibe.visipedia.org

<sup>4</sup>ebird.org



Figure 5.2: Two species of hawks from the NABirds dataset are separated into 6 categories based on their visual attributes.

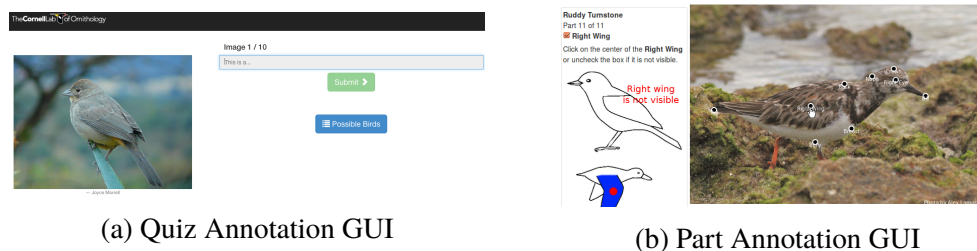


Figure 5.3: **(a)** This interface was used to collect category labels on images. Users could either use the autocomplete box or scroll through a gallery of possible birds. **(b)** This interface was used to collect part annotations on the images. Users were asked to mark the visibility and location of 11 parts. See Section 5.4 and 5.4

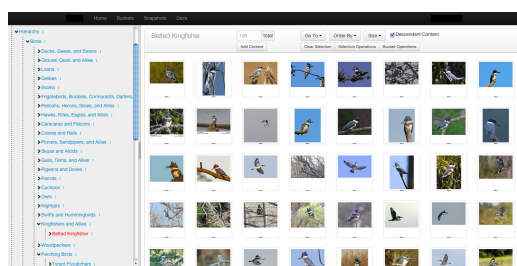


Figure 5.4: Expert interface for rapid and efficient curation of images, and easy modification of the taxonomy. The taxonomy is displayed on the left and is similar to a file system structure. See Section 5.4.

## Citizen Scientist Turkers

This is a large, passionate segment of the community motivated to help their cause. This segment is not necessarily as skilled in difficult identification tasks, but they are capable of assisting in other ways. Figure 5.3b shows a part annotation task that we deployed to this segment. The task was to simply click on all parts of the bird. The size of this population should not be underestimated. Depending on how these communities are reached, this population could be larger than the audience reached in typical crowdsourcing platforms.

## 5.5 NABirds

We used a combination of experts, citizen scientists, and MTurkers to build NABirds, a new bird dataset of 555 categories with a total of 48,562 images. Members from the birding community provided the images, the experts of the community curated the images, and a combination of CTurkers and MTurkers annotated 11 bird parts on every image along with bounding boxes. This dataset is free to use for the research community.

The taxonomy for this dataset contains 1011 nodes, and the categories cover the most common North American birds. These leaf categories were specifically chosen to allow for the creation of bird identification tools to help novice birders. Improvements on classification or detection accuracy by vision researchers will have a straightforward and meaningful impact on the birding community and their identification tools.

We used techniques from (Branson et al., 2014) to baseline performance on this dataset. Using Caffe and the fc6 layer features extracted from the entire image, we achieved an accuracy of 35.7%. Using the best performing technique from (Branson et al., 2014) with ground truth part locations, we achieved an accuracy of 75%.

## 5.6 Annotator Comparison

In this section, we compare annotations performed by Amazon Mechanical Turk workers (MTurkers) with citizen scientists reached through the Lab of Ornithology’s Facebook page. The goal of these experiments was to quantify the followings aspects of annotation tasks: (1) **Annotation Error**: The fraction of incorrect annotations; (2) **Annotation Time**: The average amount of human time required per annotation; (3) **Annotation Cost**: The average cost in dollars required per annotation; (4) **Annotation Throughput**: The average number of annotations obtainable per second, this scales with the total size of the pool of annotators.

In order to compare the skill levels of different annotator groups directly, we chose a common user interface that we considered to be appropriate for both citizen scientists and MTurkers. For category labeling tasks, we used the quiz tool that was discussed in Section 5.4. Each question presented the user with an image of a bird and requested the species label. To make the task feasible for MTurkers, we allowed users to browse through galleries of each possible species and limited the space of possible answers to  $< 40$  categories. Each quiz was focused on a particular group of birds, either sparrows or shorebirds. Random chance was  $1 / 37$  for the sparrows and  $1 / 32$  for the shorebirds. At the end of the quiz, users were given a score (the number of correct answers) and could view their results. Figure 5.3a shows our interface. We targeted the citizen scientist experts by posting the quizzes on the eBird Facebook page.

Figure 5.5 shows the distribution of scores achieved by the two different worker groups on the two different bird groups. Not surprisingly, citizen scientists had better performance on the classification task than MTurkers; however we were

uncertain as to whether or not averaging a large number of MTurkers could yield comparable performance. Figure 5.6a plots the time taken to achieve a certain error rate by combining multiple annotators for the same image using majority voting. From this figure, we can see that citizen scientists not only have a lower median time per image (about 8 seconds vs. 19 seconds), but that one citizen scientist expert label is more accurate than the average of 10 MTurker labels. We note that we are using a simple-as-possible (but commonly used) crowdsourcing method, and the performance of MTurkers could likely be improved by more sophisticated techniques such as CUBAM (Welinder, Branson, et al., 2010). However, the magnitude of difference in the two groups and overall large error rate of MTurkers led us to believe that the problem could not be solved simply using better crowdsourcing models.

Figure 5.6c measures the raw throughput of the workers, highlighting the size of the MTurk worker pool. With citizen scientists, we noticed a spike of participation when the annotation task was first posted on Facebook, and then a quick tapering off of participation. Finally, Figure 5.6b measures the cost associated with the different levels of error; citizen scientists were unpaid.

We performed a similar analysis with part annotations. For this task, we used the tool shown in Figure 5.3b. Workers from the two different groups were given an image and asked to specify the visibility and position of 11 different bird parts. We targeted the citizen scientist Turkers with this task by posting the tool on the Lab of Ornithology's Facebook page. The interface for the tool was kept the same between the workers. Figures 5.7a, 5.7b, and 5.7c detail the results of this test. From Figure 5.7a, we can see there is not a difference between the obtainable quality from the two worker groups, and that MTurkers tended to be faster at the task. Figure 5.7c again reveals that the raw throughput of MTurkers is larger than that of the citizen scientists. The primary benefit of using citizen scientists for this particular case is made clear by their zero cost in Figure 5.7b.

## Summary

From these results, we can see that there are clear distinctions between the two different worker pools. Citizen scientists are clearly more capable at labeling fine-grained categories than MTurkers. However, the raw throughput of MTurk means that you can finish annotating your dataset sooner than when using citizen scientists. If the annotation task does not require much domain knowledge (such as part

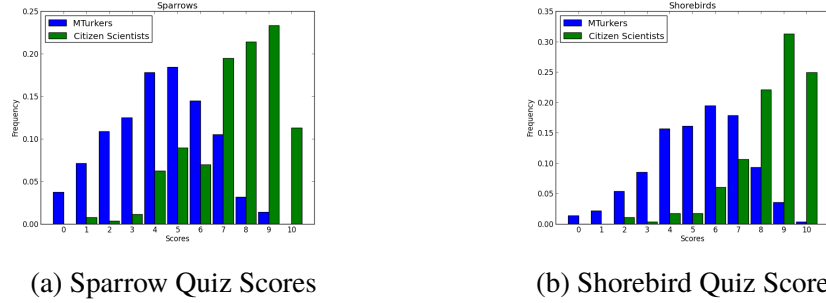


Figure 5.5: Histogram of quiz scores. Each quiz has 10 images, a perfect score is 10. **(a)** Score distributions for the sparrow quizzes. Random chance per image is 2.7%. **(b)** Score distributions for the shorebird quizzes. Random chance per image is 3.1%. See Section 5.6.

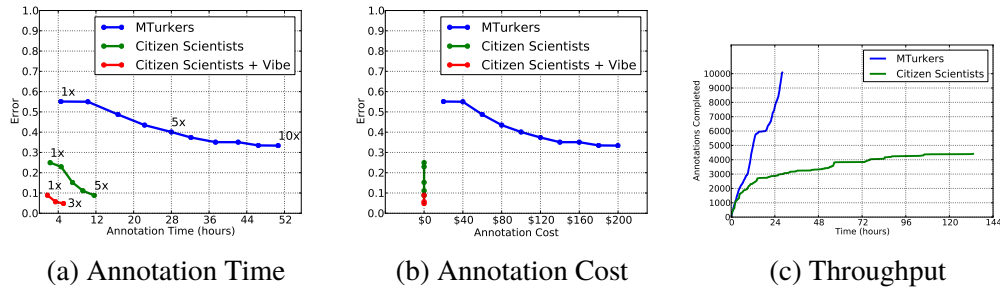


Figure 5.6: **Category Labeling Tasks:** workers used the quiz interface (see Figure 5.3a) to label the species of birds in images. **(a)** Citizen scientists are more accurate and faster for each image than MTurkers. If the citizen scientists use an expert interface (Vibe), then they are even faster and more accurate. **(b)** Citizen scientists are not compensated monetarily, they donate their time to the task. **(c)** The total throughput of MTurk is still greater, meaning you can finish annotating your dataset sooner, however this comes at a monetary cost. See Section 5.6.

annotation), then MTurkers can perform on par with citizen scientists. Gathering fine-grained category labels with MTurk should be done with care, as we have shown that naive averaging of labels does not converge to the correct label. Finally, the cost savings of using citizen scientists can be significant when the number of annotation tasks grows.

## 5.7 Measuring the Quality of Existing Datasets

CUB-200-2011 (Wah et al., 2011) and ImageNet (J. Deng et al., 2009) are two popular datasets with fine-grained categories. Both of these datasets were collected by downloading images from web searches and curating them with Amazon Mechanical Turk. Given the results in the previous section, we were interested in analyzing

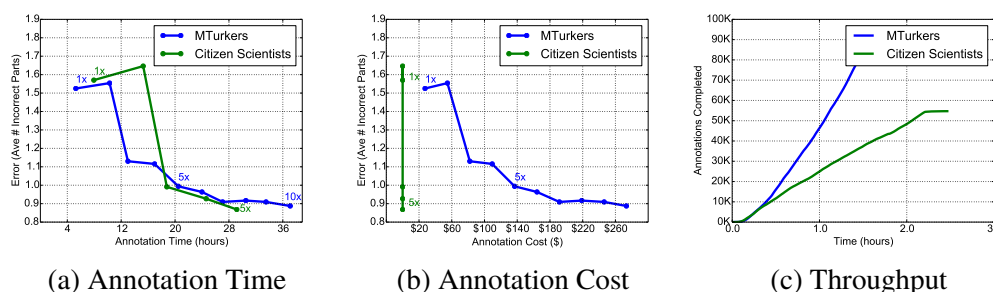


Figure 5.7: **Parts annotation tasks:** workers used the interface in Figure 5.3b to label the visibility and location of 11 parts. **(a)** For this task, as opposed to the category labeling task, citizen scientists and MTurkers perform comparable on individual images. **(b)** Citizen scientists donate their time and are not compensated monetarily. **(c)** The raw throughput of MTurk is greater than that of the citizen scientists, meaning you can finish your total annotation tasks sooner, but this comes at a cost. See Section 5.6.

the errors present in these datasets. With the help of experts from the Cornell Lab of Ornithology, we examined these datasets, specifically the bird categories, for false positives.

### CUB-200-2011:

The CUB-200-2011 dataset has 200 classes, each with roughly 60 images. Experts went through the entire dataset and identified a total of 494 errors, about 4.4% of the entire dataset. There was a total of 252 images that did not belong in the dataset because their category was not represented, and a total of 242 images that needed to be moved to existing categories. Beyond this 4.4% percent error, an additional potential concern comes from dataset bias issues. CUB was collected by performing a Flickr image search for each species, then using MTurkers to filter results. A consequence is that the most difficult images tended to be excluded from the dataset altogether. By having experts annotate the raw Flickr search results, we found that on average 11.6% of correct images of each species were incorrectly filtered out of the dataset. See Section 5.8 for additional analysis.

### ImageNet:

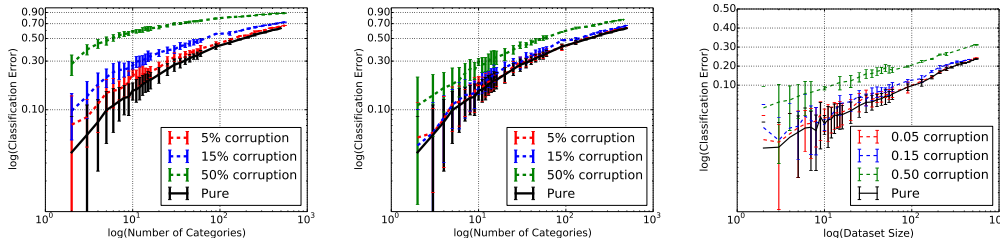
There are 59 bird categories in ImageNet, each with about 1300 images in the training set. Table 5.1 shows the false positive counts for a subset of these categories. In addition to these numbers, it was our general impression that error rate of ImageNet is probably at least as high as CUB-200 within fine-grained categories; for example, the synset “ruffed grouse, partridge, Bonasa umbellus” had overlapping definition

and image content with the synset “partridge” beyond what was quantified in our analysis.

Category	Training Images	False Positives
magpie	1300	11
kite	1294	260
dowitcher	1298	70
albatross, mollymark	1300	92
quail	1300	19
ptarmigan	1300	5
ruffed grouse, partridge, Bonasa umbellus	1300	69
prairie chicken, prairie grouse, prairie fowl	1300	52
partridge	1300	55

Table 5.1: False positives from ImageNet LSVRC dataset.

## 5.8 Effect of Annotation Quality & Quantity



(a) Image level features, train+test corruption (b) Image level features, train corruption only (c) Localized features, train corruption only

Figure 5.8: Analysis of error degradation with corrupted training labels: **(a)** Both the training and testing sets are corrupted. There is a significant difference when compared to the clean data. **(b)** Only the training set is corrupted. The induced classification error is much less than the corruption level. **(c)** Only the training set is corrupted but more part localized features are utilized. The induced classification error is still much less than the corruption level. See Section 5.8

In this section, we analyze the effect of data quality and quantity on learned vision systems. Does the 4%+ error in CUB and ImageNet actually matter? We begin with simulated label corruption experiments to quantify reduction in classification accuracy for different levels of error in Section 5.8, then perform studies on real corrupted data using an expert-vetted version of CUB in Section 5.8.



## Label Corruption Experiments

In this experiment, we attempted to measure the effect of dataset quality by corrupting the image labels of the NABirds dataset. We speculated that if an image of true class  $X$  is incorrectly labeled as class  $Y$ , the effect might be larger if class  $Y$  is included as a category in the dataset (i.e., CUB and ImageNet include only a small subset of real bird species). We thus simulated class subsets by randomly picking  $N \leq 555$  categories to comprise our sample dataset. Then, we randomly sampled  $M$  images from the  $N$  selected categories and corrupted these images by swapping their labels with another image randomly selected from all 555 categories of the original NABirds dataset. We used this corrupted dataset of  $N$  categories to build a classifier. Note that as the number of categories  $N$  within the dataset increases, the probability that a corrupted label is actually in the dataset increases. Figure 5.8 plots the results of this experiment for different configurations. We summarize our conclusions below.

### 5-10% Training error was tolerable

Figures 5.8b and 5.8c analyze the situation where only the training set is corrupted, and the ground truth testing set remains pure. We see that the increase in classification error due to 5% and even 15% corruption is remarkably low—much smaller than 5% and 15%. This result held regardless of the number of classes or computer vision algorithm. This suggests that the level of annotation error in CUB and ImageNet ( $\approx 5\%$ ) might not be a big deal.

### Obtaining a clean test set was important

On the other hand, one cannot accurately measure the performance of computer vision algorithms without a high quality test set, as demonstrated in Figure 5.8a, which measures performance when the test set is also corrupted. There is clearly a significant drop in performance with even 5% corruption. This highlights a potential problem with CUB and ImageNet, where train and test sets are equally corrupted.

### Effect of computer vision algorithm

Figure 5.8b uses computer vision algorithms based on raw image-level CNN-fc6 features (obtaining an accuracy of 35% on 555 categories) while Figure 5.8c uses a more sophisticated method (Branson et al., 2014) based on pose normalization and features from multiple CNN layers (obtaining an accuracy of 74% on 555 categories). Label corruption caused similar additive increases in test error for both

methods; however, this was a much higher percentage of the total test error for the higher performing method.

### Error Analysis on Real CUB-200-2011 Labels

The results from the previous section were obtained on simulated label corruptions. We performed additional analysis on real annotation errors on CUB-200-2011. CUB-200-2011 was collected by performing Flickr image search queries for each species and filtering the results using votes from multiple MTurkers. We had experts provide ground truth labels for all Flickr search results on 40 randomly selected categories. In Figure 5.9, we compare different possible strategies for constructing a training set based on thresholding the number of MTurk votes. Each method resulted in a different training set size and level of precision and recall. For each training set, we measured the accuracy of a computer vision classifier on a common, expert-vetted test set. The classifier was based on CNN-fc6 features from bounding box regions. Results are summarized below:

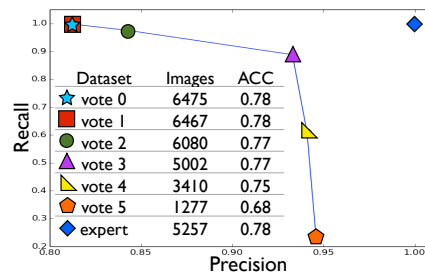


Figure 5.9: Different datasets can be built up when modifying the MTurker agreement requirement. Increasing the agreement requirement results in a dataset with low numbers of false positives and lower amounts of training data due to a high number of false negatives. A classifier trained on all the images performs as well or better than the datasets that attempt to clean up the data. See Section 5.8.

### The level of training error in CUB was tolerable

The results were consistent with those predicted by the simulated label corruption experiments, where a 5-15% error rate in the training errors yielded only a very small (roughly 1%) increase in test error. This provides comfort that CUB-200-2011 and ImageNet are still useful despite label errors. We emphasize though that an error free test set is still necessary—this is still an advantage of NABirds over CUB and ImageNet.

### Keeping all Flickr images without any MTurk curation does surprisingly well

This “free dataset” was as good as the expert dataset and slightly better than the MTurk curated datasets. The raw Flickr image search results had a reasonably high precision of 81%. Keeping all images resulted in more training images than the MTurk and expert filtered datasets. If we look at the voter agreement and the corresponding dataset training sizes, we see that having high MTurk agreement results in much smaller training set sizes and a correspondingly low recall.

### Quantity can be more important than quality

This underlines the point that having a large training set is extremely important, and having strict requirements on annotation quality can come at the expense of reducing training set size. We randomly reduced the size of the training set within the 40 class dataset and measured performance of each resulting computer vision classifier. The results are shown in Table 5.2; we see that classification accuracy is more sensitive to training set size than it was to label corruption (see Figures 5.8b and 5.9).

Scale Size	1	1/2	1/4	1/8	1/16	1/32	1/64
ACC	.77	.73	.676	.612	.517	.43	.353

Table 5.2: Classification accuracy with reduced training set size. See Section 5.8.

### Similar results when scaling to more classes

One caveat is that the above results were obtained on a 40 class subset, which was the limit of what was reasonable to ask of experts to annotate all Flickr image search results. It is possible that annotation quality becomes more important as the number of classes in the dataset grows. To test this, we had experts go through all 200 classes in CUB-200-2011, annotating all images that were included in the dataset (see Section 5.7). We obtained a similar result as on the 40-class subset, where the expert filtered dataset performed at about the same level as the original CUB-200-2011 trainset that contains 4-5% error. These results are consistent with simulated label corruption experiments in Figure 5.8b.

## 5.9 Conclusion

We introduced tools for crowdsourcing computer vision annotations using citizen scientists. In collecting a new expert-curated dataset of 48,562 images over 555 categories, we found that citizen scientists provide significantly higher quality la-

bels than Mechanical Turk workers, and found that Turkers have alarmingly poor performance annotating fine-grained classes. This has resulted in error rates of over 4% in fine-grained categories in popular datasets like CUB-200-2011 and ImageNet. Despite this, we found that learning algorithms based on CNN features and part localization were surprisingly robust to mislabeled training examples as long as the error rate is not too high, and we would like to emphasize that ImageNet and CUB-200-2011 are still very useful and relevant datasets for research in computer vision.

Our results so far have focused on experiences in a single domain (birds) and have resulted in a new publicly available tool for bird species identification. We are currently working on expanding to other types of categories such as shoes and Lepidoptera. Given that over 500 citizen scientists helped provide high quality annotations in just a single domain, working with citizen scientists has potential to generate datasets of unprecedented size and quality while encouraging the landscape of computer vision research to shape around the interests of end users.

### 5.10 Acknowledgments

We would like to thank Nathan Goldberg, Ben Barkley, Brendan Fogarty, Graham Montgomery, and Nathaniel Hernandez for assisting with the user experiments. We appreciate the feedback and general guidance from Miyoko Chu, Steve Kelling, Chris Wood, and Alex Chang. This work was supported in part by a Google Focused Research Award, the Jacobs Technion-Cornell Joint Research Fund, and Office of Naval Research MURI N000141010933.

### References

- Angelova, Anelia and Shenghuo Zhu (2013). “Efficient Object Detection and Segmentation for Fine-Grained Recognition”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Berg, Tamara L, Alexander C Berg, and Jonathan Shih (2010). “Automatic attribute discovery and characterization from noisy web data”. In: *European Conference on Computer Vision*. Springer, pp. 663–676.
- Berg, Thomas et al. (2014). “Birdsnap: Large-Scale Fine-Grained Visual Categorization of Birds”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 2019–2026. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.259. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6909656>.

- Boom, Bastiaan J. et al. (2014). “A research tool for long-term and continuous analysis of fish assemblage in coral-reefs using underwater camera footage”. In: *Ecological Informatics* 23, pp. 83–97. ISSN: 15749541. DOI: 10.1016/j.ecoinf.2013.10.006. URL: <http://www.sciencedirect.com/science/article/pii/S1574954113001003>.
- Branson, Steve et al. (2014). “Bird Species Categorization Using Pose Normalized Deep Convolutional Nets”. In: *arXiv preprint arXiv:1406.2952*.
- Deng, Jia, Wei Dong, et al. (2009). “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pp. 248–255.
- Deng, Jia, Jonathan Krause, and Li Fei-Fei (2013). “Fine-grained crowdsourcing for fine-grained recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587.
- Deng, J. et al. (2009). “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*.
- Everingham, M. et al. (2010). “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2, pp. 303–338.
- Fei-Fei, Li, Robert Fergus, and Pietro Perona (2006). “One-shot learning of object categories”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.4, pp. 594–611.
- Girshick, Ross et al. (2013). “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *arXiv preprint arXiv:1311.2524*.
- Gneezy, Uri and Aldo Rustichini (2000). “Pay enough or don’t pay at all”. In: *Quarterly journal of economics*, pp. 791–810.
- Griffin, G, A Holub, and P Perona (2007). *Caltech-256 Object Category Dataset*. Tech. rep. CNS-TR-2007-001. California Institute of Technology. URL: <http://authors.library.caltech.edu/7694>.
- Ipeirotis, Panagiotis G. and Evgeniy Gabrilovich (2014). “Quizz: targeted crowdsourcing with a billion (potential) users”. In: pp. 143–154. DOI: 10.1145/2566486.2567988. URL: <http://dl.acm.org/citation.cfm?id=2566486.2567988>.
- Ipeirotis, Panagiotis G., Foster Provost, et al. (2013). “Repeated labeling using multiple noisy labelers”. In: *Data Mining and Knowledge Discovery* 28.2, pp. 402–441. ISSN: 1384-5810. DOI: 10.1007/s10618-013-0306-1. URL: <http://link.springer.com/10.1007/s10618-013-0306-1>.
- Khosla, Aditya et al. (2011). “Novel Dataset for Fine-Grained Image Categorization”. In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO.

- Krause, Jonathan et al. (2013). “Collecting a Large-Scale Dataset of Fine-Grained Cars”. In: *Second Workshop on Fine-Grained Visual Categorization (FGVC2)*.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks.” In: *NIPS*.
- Kuznetsov, Stacey (2006). “Motivations of contributors to Wikipedia”. In: *ACM SIGCAS computers and society* 36.2, p. 1.
- Lazebnik, S., C. Schmid, and Jean Ponce (2004). “Semi-Local Affine Parts for Object Recognition”. In: *Proc. BMVC*. doi:10.5244/C.18.98, pp. 98.1–98.10. ISBN: 1-901725-25-1.
- Lin, Tsung-Yi et al. (2014). “Microsoft COCO: Common objects in context”. In: *ECCV*.
- Liu, Jiongxin et al. (2012). “Dog Breed Classification Using Part Localization.” In: *ECCV*.
- Long, Chengjiang, Gang Hua, and Ashish Kapoor (2013). “Active Visual Recognition with Expertise Estimation in Crowdsourcing”. In: *2013 IEEE International Conference on Computer Vision*. IEEE, pp. 3000–3007. ISBN: 978-1-4799-2840-8. DOI: 10.1109/ICCV.2013.373. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6751484>.
- Maji, S. et al. (2013). *Fine-Grained Visual Classification of Aircraft*. Tech. rep. arXiv: 1306.5151 [cs-cv].
- Martinez-Munoz, G. et al. (2009). “Dictionary-free categorization of very similar objects via stacked evidence trees”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 549–556. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206574. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5206574>.
- Nilsback, M-E. and A. Zisserman (2008). “Automated Flower Classification over a Large Number of Classes”. In: *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*.
- Nov, Oded (2007). “What motivates wikipedians?” In: *Communications of the ACM* 50.11, pp. 60–64.
- Parkhi, Omkar M et al. (2011). “The truth about cats and dogs”. In: *ICCV*.
- Raykar, Vikas C et al. (2009). “Supervised learning from multiple experts: whom to trust when everyone lies a bit”. In: *Proceedings of the 26th Annual international conference on machine learning*. ACM, pp. 889–896.
- Russakovsky, Olga et al. (2014). *ImageNet Large Scale Visual Recognition Challenge*. eprint: arXiv:1409.0575.

- Sheng, Victor S., Foster Provost, and Panagiotis G. Ipeirotis (2008). “Get another label? improving data quality and data mining using multiple, noisy labelers”. In: *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*. New York, New York, USA: ACM Press, p. 614. ISBN: 9781605581934. DOI: 10.1145/1401890.1401965. URL: <http://dl.acm.org/citation.cfm?id=1401890.1401965>.
- Sorokin, Alexander and David Forsyth (2008). “Utility data annotation with Amazon Mechanical Turk”. In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, pp. 1–8. ISBN: 978-1-4244-2339-2. DOI: 10.1109/CVPRW.2008.4562953. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4562953>.
- Taigman, Yaniv et al. (2014). “Deepface: Closing the gap to human-level performance in face verification”. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, pp. 1701–1708.
- Von Ahn, Luis (2006). “Games with a purpose”. In: *Computer* 39.6, pp. 92–94.
- Von Ahn, Luis et al. (2008). “recaptcha: Human-based character recognition via web security measures”. In: *Science* 321.5895, pp. 1465–1468.
- Wah, C. et al. (2011). *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep. CNS-TR-2011-001. California Institute of Technology.
- Welinder, Peter, Steve Branson, et al. (2010). “The Multidimensional Wisdom of Crowds”. In: *Advances in Neural Information Processing Systems 23*. Ed. by J D Lafferty et al. Curran Associates, Inc., pp. 2424–2432. URL: <http://papers.nips.cc/paper/4074-the-multidimensional-wisdom-of-crowds.pdf>.
- Welinder, Peter and Pietro Perona (2010). “Online crowdsourcing: Rating annotators and obtaining cost-effective labels”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*. IEEE, pp. 25–32. ISBN: 978-1-4244-7029-7. DOI: 10.1109/CVPRW.2010.5543189. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5543189>.
- Whitehill, Jacob et al. (2009). “Whose vote should count more: Optimal integration of labels from labelers of unknown expertise”. In: *Advances in neural information processing systems*, pp. 2035–2043.
- Yang, Heng-Li and Cheng-Yu Lai (2010). “Motivations of Wikipedia content contributors”. In: *Computers in Human Behavior* 26.6, pp. 1377–1383.

*Chapter 6*THE INATURALIST SPECIES CLASSIFICATION AND  
DETECTION DATASET

Van Horn, Grant et al. (2018). “The iNaturalist Species Classification and Detection Dataset”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT. doi: 10.1109/CVPR.2018.00914.

**6.1 Abstract**

Existing image classification datasets used in computer vision tend to have a uniform distribution of images across object categories. In contrast, the natural world is heavily imbalanced, as some species are more abundant and easier to photograph than others. To encourage further progress in challenging real world conditions, we present the iNaturalist species classification and detection dataset, consisting of 859,000 images from over 5,000 different species of plants and animals. It features visually similar species captured in a wide variety of situations, from all over the world. Images were collected with different camera types, have varying image quality, feature a large class imbalance, and have been verified by multiple citizen scientists. We discuss the collection of the dataset and present extensive baseline experiments using state-of-the-art computer vision classification and detection models. Results show that current non-ensemble based methods achieve only 67% top one classification accuracy, illustrating the difficulty of the dataset. Specifically, we observe poor results for classes with small numbers of training examples, suggesting more attention is needed in low-shot learning.

**6.2 Introduction**

Performance on existing image classification benchmarks such as (Russakovsky et al., 2015) is close to being saturated by the current generation of classification algorithms (He et al., 2016; Szegedy, Vanhoucke, et al., 2016; Szegedy, Ioffe, et al., 2016; Xie et al., 2017). However, the number of training images is crucial. If one reduces the number of training images per category, performance typically suffers. It may be tempting to try and acquire more training data for the classes with few images, but this is often impractical, or even impossible, in many application domains. We argue that class imbalance is a property of the real world, and computer vision



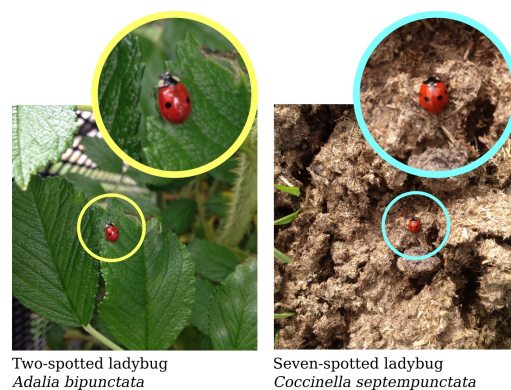


Figure 6.1: Two visually similar species from the iNat2017 dataset. Through close inspection, we can see that the ladybug on the left has *two* spots while the one on the right has *seven*.

models should be able to deal with it. Motivated by this problem, we introduce the iNaturalist Classification and Detection Dataset (iNat2017). Just like the real world, it exhibits a large class imbalance, as some species are much more likely to be observed.

It is estimated that the natural world contains several million species with around 1.2 million of these having already been formally described (Mora et al., 2011). For some species, it may only be possible to determine the species via genetics or by dissection. For the rest, visual identification in the wild, while possible, can be extremely challenging. This can be due to the sheer number of visually similar categories that an individual would be required to remember along with the challenging inter-class similarity; see Fig. 6.1. As a result, there is a critical need for robust and accurate automated tools to scale up biodiversity monitoring on a global scale (Cardinale et al., 2012).

The iNat2017 dataset is comprised of images and labels from the citizen science website iNaturalist<sup>1</sup>. The site allows naturalists to map and share photographic observations of biodiversity across the globe. Each observation consists of a date, location, images, and labels containing the name of the species present in the images. As of November 2017, iNaturalist has collected over 6.6 million observations from 127,000 species. From this, there are close to 12,000 species that have been observed by at least twenty people and have had their species ID confirmed by multiple annotators.

The goal of iNat2017 is to push the state-of-the-art in image classification and

---

<sup>1</sup>[www.inaturalist.org](http://www.inaturalist.org)

detection for ‘in the wild’ data featuring large numbers of imbalanced, fine-grained, categories. iNat2017 contains over 5,000 species, with a combined training and validation set of 675,000 images, 183,000 test images, and over 560,000 manually created bounding boxes. It is free from one of the main selection biases that are encountered in many existing computer vision datasets - as opposed to being scraped from the web, all images have been collected and then verified by multiple citizen scientists. It features many visually similar species captured in a wide variety of situations from all over the world. We outline how the dataset was collected and report extensive baseline performance for state-of-the-art classification and detection algorithms. Our results indicate that iNat2017 is challenging for current models due to its imbalanced nature and will serve as a good experimental platform for future advances in our field.

### 6.3 Related Datasets

In this section, we review existing image classification datasets commonly used in computer vision. Our focus is on large scale, fine-grained, object categories as opposed to datasets that feature common everyday objects, e.g. (Fei-Fei, Fergus, and Perona, 2007; Everingham et al., 2010; T.-Y. Lin et al., 2014). Fine-grained classification problems typically exhibit two distinguishing differences from their coarse-grained counter parts. First, there tends to be only a small number of domain experts that are capable of making the classifications. Second, as we move down the spectrum of granularity, the number of instances in each class becomes smaller. This motivates the need for automated systems that are capable of discriminating between large numbers of potentially visually similar categories with small numbers of training examples for some categories. In the extreme, face identification can be viewed as an instance of fine-grained classification, and many existing benchmark datasets with long tail distributions exist e.g. (G. B. Huang et al., 2007; Omkar M Parkhi, Vedaldi, Zisserman, et al., 2015; Guo et al., 2016; Cao et al., 2017). However, due to the underlying geometric similarity between faces, current state-of-the-art approaches for face identification tend to perform a large amount of face specific pre-processing (Taigman et al., 2014; Schroff, Kalenichenko, and Philbin, 2015; Omkar M Parkhi, Vedaldi, Zisserman, et al., 2015).

The vision community has released many fine-grained datasets covering several domains such as birds (Welinder et al., 2010; Wah et al., 2011; Berg et al., 2014; Van Horn et al., 2015; Krause, Sapp, et al., 2016), dogs (Khosla et al., 2011; O. M. Parkhi et al., 2012; J. Liu et al., 2012), airplanes (Maji et al., 2013; Vedaldi et al.,

2014), flowers (Nilsback and Zisserman, 2006), leaves (Kumar et al., 2012), food (Hou, Y. Feng, and Wang, 2017), trees (Wegner et al., 2016), and cars (Krause, Stark, et al., 2013; Y.-L. Lin et al., 2014; Yang et al., 2015; Gebru et al., 2017). ImageNet (Russakovsky et al., 2015) is not typically advertised as a fine-grained dataset, yet it contains several groups of fine-grained classes, including about 60 bird species and about 120 dog breeds. In Table 6.1, we summarize the statistics of some of the most common datasets. With the exception of a small number e.g.(Krause, Sapp, et al., 2016; Gebru et al., 2017), many of these datasets were typically constructed to have an approximately uniform distribution of images across the different categories. In addition, many of these datasets were created by searching the internet with automated web crawlers and as a result, can contain a large proportion of incorrect images e.g.(Krause, Sapp, et al., 2016). Even manually vetted datasets such as ImageNet (Russakovsky et al., 2015) have been reported to contain up to 4% error for some fine-grained categories (Van Horn et al., 2015). While current deep models are robust to label noise at training time, it is still very important to have clean validation and test sets to be able to quantify performance (Van Horn et al., 2015; Rolnick et al., 2017).

Unlike web scraped datasets (Krause, Sapp, et al., 2016; Krasin et al., 2016; Wilber et al., 2017; Hou, Y. Feng, and Wang, 2017), the annotations in iNat2017 represent the consensus of informed enthusiasts. Images of natural species tend to be challenging, as individuals from the same species can differ in appearance due to sex and age and may also appear in different environments. Depending on the particular species, they can also be very challenging to photograph in the wild. In contrast, mass-produced, man-made object categories are typically identical up to nuisance factors, i.e. they only differ in terms of pose, lighting, or color, but not necessarily in their underlying object shape or appearance (Yu and Grauman, 2014; Gebru et al., 2017; Zhang et al., 2017).

## 6.4 Dataset Overview

In this section, we describe the details of the dataset, including how we collected the image data (Section 6.4), how we constructed the train, validation and test splits (Section 6.4), how we vetted the test split (Section 6.4), and how we collected bounding boxes (Section 6.4). Future researchers may find our experience useful when constructing their own datasets.

Dataset Name	# Train	# Classes	Imbalance
Flowers 102 (Nilsback and Zisserman, 2006)	1,020	102	1.00
Aircraft (Maji et al., 2013)	3,334	100	1.03
Oxford Pets (O. M. Parkhi et al., 2012)	3,680	37	1.08
DogSnap (J. Liu et al., 2012)	4,776	133	2.85
CUB 200-2011 (Wah et al., 2011)	5,994	200	1.03
Stanford Cars (Krause, Stark, et al., 2013)	8,144	196	2.83
Stanford Dogs (Khosla et al., 2011)	12,000	120	1.00
Urban Trees (Wegner et al., 2016)	14,572	18	7.51
NABirds (Van Horn et al., 2015)	23,929	555	15.00
LeafSnap* (Kumar et al., 2012)	30,866	185	8.00
CompCars* (Yang et al., 2015)	136,727	1,716	10.15
VegFru* (Hou, Y. Feng, and Wang, 2017)	160,731	292	8.00
Census Cars (Gebru et al., 2017)	512,765	2,675	10.00
ILSVRC2012 (Russakovsky et al., 2015)	1,281,167	1,000	1.78
<b>iNat2017</b>	<b>579,184</b>	<b>5,089</b>	<b>435.44</b>

Table 6.1: Summary of popular general and fine-grained computer vision classification datasets. ‘Imbalance’ represents the number of images in the largest class divided by the number of images in the smallest. While susceptible to outliers, it gives an indication of the imbalance found in many common datasets. \*Total number of train, validation, and test images.

### Dataset Collection

iNat2017 was collected in collaboration with iNaturalist, a citizen science effort that allows naturalists to map and share observations of biodiversity across the globe through a custom made web portal and mobile apps. Observations, submitted by *observers*, consist of images, descriptions, location and time data, and community identifications. If the community reaches a consensus on the taxa in the observation, then a “research-grade” label is applied to the observation. iNaturalist makes an archive of research-grade observation data available to the environmental science community via the Global Biodiversity Information Facility (GBIF) (Ueda, 2017). Only research-grade labels at genus, species, or lower are included in this archive. These archives contain the necessary information to reconstruct which photographs belong to each observation, which observations belong to each observer, as well as the taxonomic hierarchy relating the taxa. These archives are refreshed on a rolling basis, and the iNat2017 dataset was created by processing the archive from October 3rd, 2016.














	Super-Class	Class	Train	Val	BBoxes
	Plantae	2,101	158,407	38,206	-
	Insecta	1,021	100,479	18,076	125,679
	Aves	964	214,295	21,226	311,669
	Reptilia	289	35,201	5,680	42,351
	Mammalia	186	29,333	3,490	35,222
	Fungi	121	5,826	1,780	-
	Amphibia	115	15,318	2,385	18,281
	Mollusca	93	7,536	1,841	10,821
	Animalia	77	5,228	1,362	8,536
	Arachnida	56	4,873	1,086	5,826
	Actinopterygii	53	1,982	637	3,382
	Chromista	9	398	144	-
	Protozoa	4	308	73	-
	<b>Total</b>	5,089	579,184	95,986	561,767

Table 6.2: Number of images, classes, and bounding boxes in iNat2017 broken down by super-class. ‘Animalia’ is a catch-all category that contains species that do not fit in the other super-classes. Bounding boxes were collected for nine of the super-classes. In addition, the public and private test sets contain 90,427 and 92,280 images, respectively.

### Dataset Construction

The complete GBIF archive had 54k classes (genus level taxa and below), with 1.1M observations and a total of 1.6M images. However, over 19k of those classes contained only one observation. In order to construct train, validation, and test splits that contained samples from all classes, we chose to employ a taxa selection criterion: we required that a taxa have at least 20 observations, submitted from at least 20 unique observers (i.e. one observation from each of the 20 unique observers). This criterion limited the candidate set to 5,089 taxa coming from 13 super-classes, see Table 6.2.

The next step was to partition the images from these taxa into the train, validation, and test splits. For each of the selected taxa, we sorted the *observers* by their number of observations (fewest first), selected the first 40% of observers to be in the test split, with the remaining 60% to be in the “train-val” split. By partitioning the observers in this way and subsequently placing all of their photographs into one split or the other, we ensure that the behavior of a particular user (e.g. camera equipment, location, background, *etc.*) is contained within a single split, and not available as a useful source of information for classification on the other split for a specific taxa. Note that a particular observer may be put in the test split for one

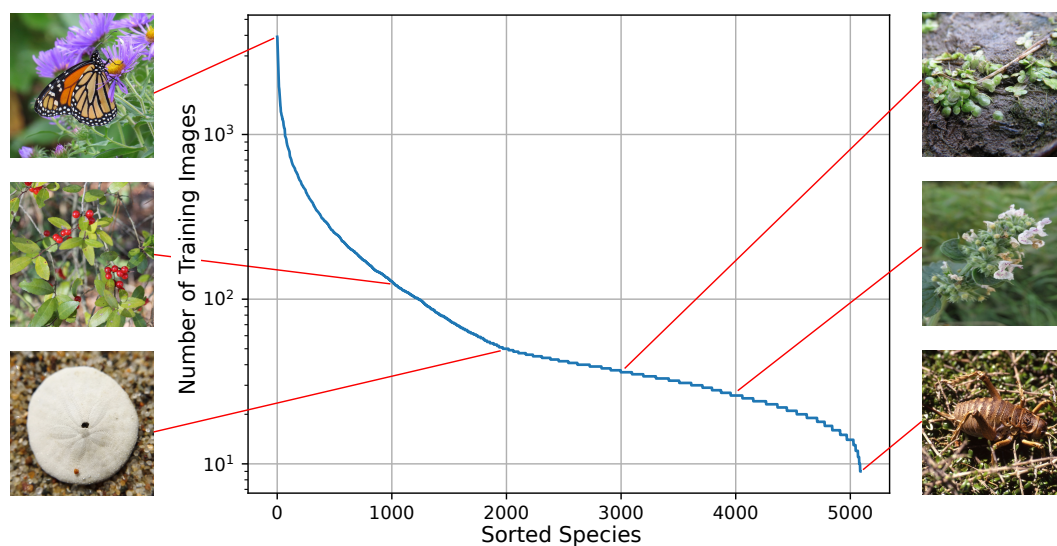


Figure 6.2: Distribution of training images per species. iNat2017 contains a large imbalance between classes, where the top 1% most populated classes contain over 16% of training images.

taxa, but the “train-val” split for another taxa. By first sorting the observers by their number of observations, we ensure that the test split contains a high number of unique observers and therefore a high degree of variability. To be concrete, at this point, for a taxa that has exactly 20 unique observers (the minimum allowed), 8 observers would be placed in the test split, and the remaining 12 observers would be placed in the “train-val” split. Rather than release all test images, we randomly sampled  $\sim 183,000$  to be included in the final dataset. The remaining test images were held in reserve in case we encountered unforeseen problems with the dataset.

To construct the separate train and validation splits for each taxa from the “train-val” split, we again partition on the observers. For each taxa, we sort the observers by increasing observation counts and repeatedly add observers to the validation split until either of the following conditions occurs: (1) the total number of *photographs* in the validation set exceeds 30, or (2) 33% of the available *photographs* in the “train-val” set for the taxa have been added to the validation set. The remaining observers and all of their photographs are added to the train split. To be concrete, and continuing the example from above, exactly 4 images would be placed in the validation split, and the remaining 8 images would be placed in the train split for a taxa with 20 unique observers. This results in a validation split that has at least 4 and at most  $\sim 30$  images for each class (the last observer added to the validation split for a taxa may push the number of photographs above 30), and a train split that has

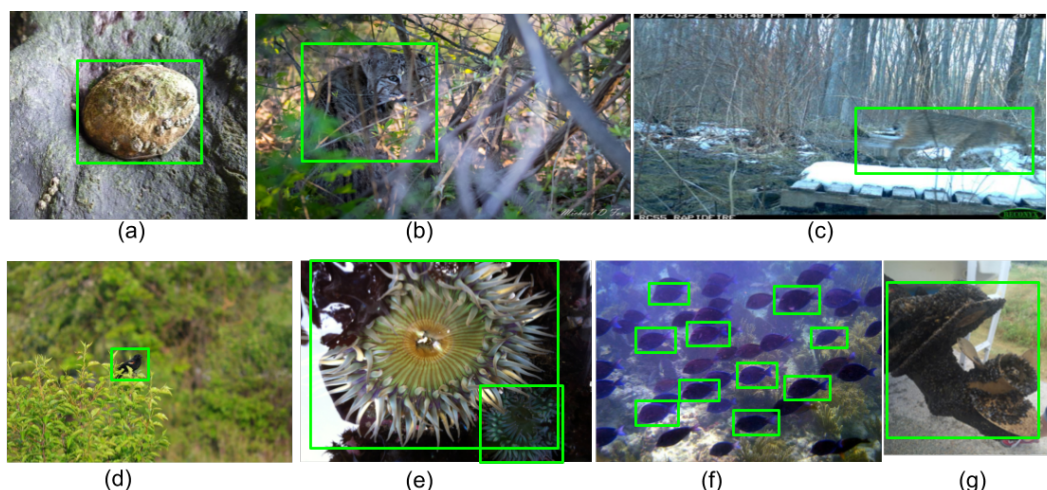


Figure 6.3: Sample bounding box annotations. Annotators were asked to annotate up to 10 instances of a super-class, as opposed to the fine-grained class, in each image.

at least 8 images for each class. See Fig. 6.2 for the distribution of train images per class.

At this point we have the final image splits, with a total of 579,184 training images, 95,986 validation images, and 182,707 test images. All images were resized to have a max dimension of 800px. Sample images from the dataset can be viewed in Fig. 6.11. The iNat2017 dataset is available from our project website<sup>2</sup>.

### Test Set Verification

Each observation on iNaturalist is made up of one or more images that provide evidence that the taxon *was present*. Therefore, a small percentage of images may not contain the taxon of interest but instead can include footprints, feces, and habitat shots. Unfortunately, iNaturalist does not distinguish between these types of images in the GBIF export, so we crowdsourced the verification of three super-classes (Mammalia, Aves, and Reptilia) that might exhibit these “non-instance” images. We found that less than 1.1% of the test set images for Aves and Reptilia had non-instance images. The fraction was higher for Mammalia due to the prevalence of footprint and feces images, and we filtered these images out of the test set. The training and validation images were not filtered.

<sup>2</sup>[https://github.com/visipedia/inat\\_comp/tree/master/2017](https://github.com/visipedia/inat_comp/tree/master/2017)



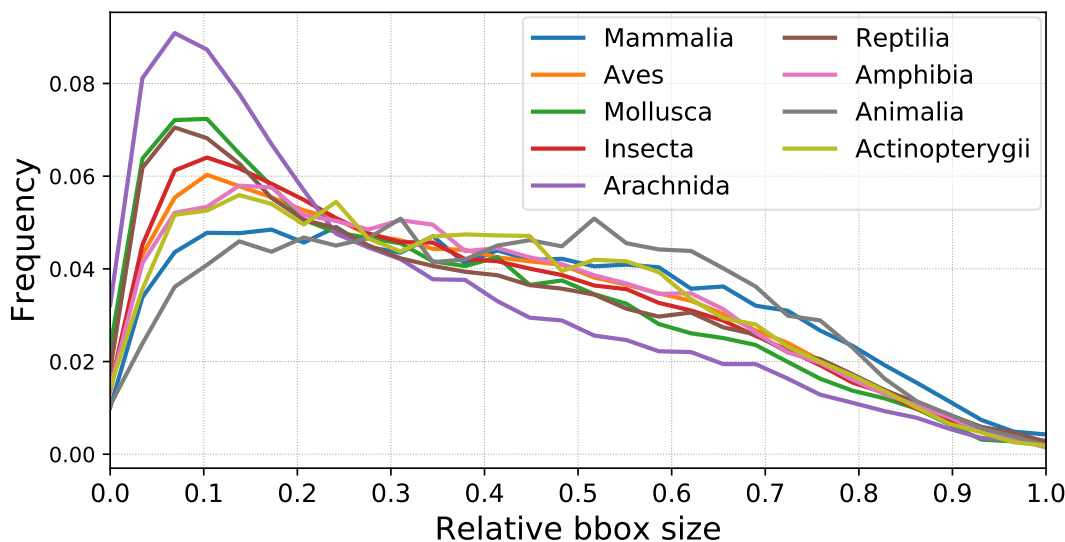


Figure 6.4: The distribution of relative bounding box sizes (calculated by  $\sqrt{w_{bbox} \times h_{bbox}} / \sqrt{w_{img} \times h_{img}}$ ) in the training set, per super-class. Most objects are relatively small or medium sized.

### Bounding Box Annotation

Bounding boxes were collected on 9 out of the 13 super-classes (see Table 6.2), totaling 2,854 classes. Due to the inherent difficulty of asking non-expert crowd annotators to both recognize and box specific fine-grained classes, we instructed annotators to instead box all instances of the associated super-class for a taxon (e.g. “Box all Birds” rather than “Box all Red-winged Black Birds”). We collected super-class boxes only on taxa that are part of that super-class. For some super-classes (e.g. Mollusca), there are images containing taxa which are unfamiliar to many of the annotators (e.g. Fig. 6.3(a)). For those cases, we instructed the annotators to box the prominent objects in the images.

The task instructions specified to draw boxes tightly around all parts of the animal (including legs, horns, antennas, *etc.*). If the animal is occluded, the annotators were instructed to draw the box around the visible parts (e.g. Fig. 6.3(b)). In cases where the animal is blurry or small (e.g. Fig. 6.3(c) and (d)), the following rule-of-thumb was used: “if you are confident that it is an animal from the requested super-class, regardless of size, blurriness or occlusion, put a box around it.” For images with multiple instances of the super-class, all of them are boxed, up to a limit of 10 (Fig. 6.3(f)), and bounding boxes may overlap (Fig. 6.3(e)). We observe that 12% of images have more than 1 instance and 1.3% have more than 5. If the instances are physically connected (e.g. the mussels in Fig. 6.3(g)), then only one box is placed around them.



Bounding boxes were not collected on the Plantae, Fungi, Protozoa, or Chromista super-classes because these super-classes exhibit properties that make it difficult to box the individual instances (e.g. close up of trees, bushes, kelp, *etc.*). An alternate form of pixel annotations, potentially from a more specialized group of crowd workers, may be more appropriate for these classes.

Under the above guidelines, 561,767 bounding boxes were obtained from 449,313 images in the training and validation sets. Following the size conventions of COCO T.-Y. Lin et al., 2014, the iNat2017 dataset is composed of 5.7% small instances ( $\text{area} < 32^2$ ), 23.6% medium instances ( $32^2 \leq \text{area} \leq 96^2$ ), and 70.7% large instances ( $\text{area} > 96^2$ ), with area computed as 50% of the annotated bounding box area (since segmentation masks were not collected). Figure 6.4 shows the distribution of relative bounding box sizes, indicating that a majority of instances are relatively small and medium sized.

## 6.5 Experiments

In this section, we compare the performance of state-of-the-art classification and detection models on iNat2017.

### Classification Results

To characterize the classification difficulty of iNat2017, we ran experiments with several state-of-the-art deep network architectures, including ResNets (He et al., 2016), Inception V3 (Szegedy, Vanhoucke, et al., 2016), Inception ResNet V2 (Szegedy, Ioffe, et al., 2016), and MobileNet (Howard et al., 2017). During training, random cropping with aspect ratio augmentation (Szegedy, W. Liu, et al., 2015) was used. Training batches of size 32 were created by uniformly sampling from all available training images as opposed to sampling uniformly from the classes. We fine-tuned all networks from ImageNet pre-trained weights with a learning rate of 0.0045, decayed exponentially by 0.94 every 4 epochs, and RMSProp optimization with momentum and decay both set to 0.9. Training and testing were performed with an image size of  $299 \times 299$ , with a single centered crop at test time.

Table 6.3 summarizes the top-1 and top-5 accuracy of the models. From the Inception family, we see that the higher capacity Inception ResNet V2 outperforms the Inception V3 network. The addition of the Squeeze-and-Excitation (SE) blocks (Hu, Shen, and G. Sun, 2017) further improves performance for both models by a small amount. ResNets performed worse on iNat2017 compared to the Inception architectures, likely due to over-fitting on categories with a small number of training

	bf Validation		Public Test		Private Test	
	Top1	Top5	Top1	Top5	Top1	Top5
IncResNetV2 SE	<b>67.3</b>	<b>87.5</b>	<b>68.5</b>	<b>88.2</b>	67.7	<b>87.9</b>
IncResNetV2	67.1	<b>87.5</b>	68.3	88.0	<b>67.8</b>	87.8
IncV3 SE	65.0	85.9	66.3	86.7	65.2	86.3
IncV3	64.2	85.2	65.5	86.1	64.8	85.7
ResNet152 drp	62.6	84.5	64.2	85.5	63.1	85.1
ResNet101 drp	60.9	83.1	62.4	84.1	61.4	83.6
ResNet152	59.0	80.5	60.6	81.7	59.7	81.3
ResNet101	58.4	80.0	59.9	81.2	59.1	80.9
MobileNet V1	52.9	75.4	54.4	76.8	53.7	76.3

Table 6.3: Classification results for various CNNs trained on only the training set, using a single center crop at test time. Unlike some current datasets where performance is near saturation, iNat2017 still poses a challenge for state-of-the-art classifiers.

images. We found that adding a 0.5 probability dropout layer (drp) could improve the performance of ResNets. MobileNet, designed to efficiently run on embedded devices, had the lowest performance.

Overall, the Inception ResNetV2 SE was the best performing model. As a comparison, this model achieves a single crop top-1 and top-5 accuracy of 80.2% and 95.21% respectively on the ILSVRC 2012 (Russakovsky et al., 2015) validation set (Szegedy, Ioffe, et al., 2016), as opposed to 67.74% and 87.89% on iNat2017, highlighting the comparative difficulty of the iNat2017 dataset. A more detailed super-class level breakdown is available in Table 6.4 for the Inception ResNetV2 SE model. We can see that the Reptilia super-class (with 289 classes) was the most difficult with an average top-1 accuracy of 45.87%, while the Protozoa super-class (with 4 classes) had the highest accuracy at 89.19%. Viewed as a collection of fine-grained datasets (one for each super-class), we can see that the iNat2017 dataset exhibits highly variable classification difficulty.

In Figure 6.5, we plot the top one public test set accuracy against the number of training images for each class from the Inception ResNet V2 SE model. We see that as the number of training images per class increases, so does the test accuracy. However, we still observe a large variance in accuracy for classes with a similar amount of training data, revealing opportunities for algorithmic improvements in both the low data and high data regimes.

Super-Class	Avg Train	Public Test	
		Top1	Top5
Plantae	75.4	69.5	87.1
Insecta	98.4	77.1	93.4
Aves	222.3	67.3	88.0
Reptilia	121.8	45.9	80.9
Mammalia	157.7	61.4	85.1
Fungi	48.1	74.0	92.3
Amphibia	67.9	51.2	81.0
Mollusca	81.0	72.4	90.9
Animalia	67.9	73.8	91.1
Arachnida	87.0	71.5	88.8
Actinopterygii	37.4	70.8	86.3
Chromista	44.2	73.8	92.4
Protozoa	77.0	89.2	96.0

Table 6.4: Super-class level accuracy (computed by averaging across all species within each super-class) for the best performing model Inception ResNetV2 SE (Hu, Shen, and G. Sun, 2017). “Avg Train” indicates the average number of training images per class for each super-class. We observe a large difference in performance across the different super-classes.

### Additional Classification Results

We performed an experiment to understand if there was any relationship between real world animal size and prediction accuracy. Using existing records for bird (Lislevand, Figuerola, and Székely, 2007) and mammal (Jones et al., 2009) body sizes, we assigned a mass to each of the classes in iNat2017 that overlapped with these datasets. For a given species, mass will vary due to the life stage or gender of the particular individual. Here, we simply take the average value. This resulted in data for 795 species, from the small Allen’s hummingbird (*Selasphorus sasin*) to the large Humpback whale (*Megaptera novaeangliae*). In Figure 6.6, we can see that median accuracy decreases as the mass of the species increases. These results are preliminary, but reinforce the observation that it can be challenging for humans to take good photographs of larger mammals. More analysis of these failure cases may allow us to produce better, species-specific, instructions for the photographers on iNaturalist.

The IUCN Red List of Vulnerable Species monitors and evaluates the extinction risk of thousands of species and subspecies (Baillie, Hilton-Taylor, and Stuart, 2004). In Figure 6.7, we plot the Red List status of 1,568 species from the iNat2017 dataset. We see that the vast majority of the species are in the ‘Least Concern’ category and that test accuracy decreases as the threatened status increases. This can perhaps be

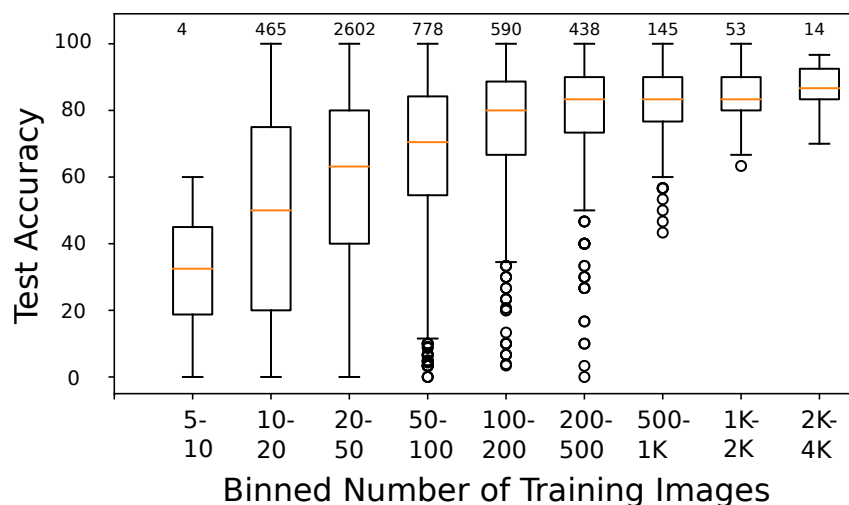


Figure 6.5: Top one public test set accuracy per class for IncResNet V2 SE (Hu, Shen, and G. Sun, 2017). Each box plot represents classes grouped by the number of training images. The number of classes for each bin is written on top of each box plot. Performance improves with the number of training images, but the challenge is how to maintain high accuracy with fewer images.

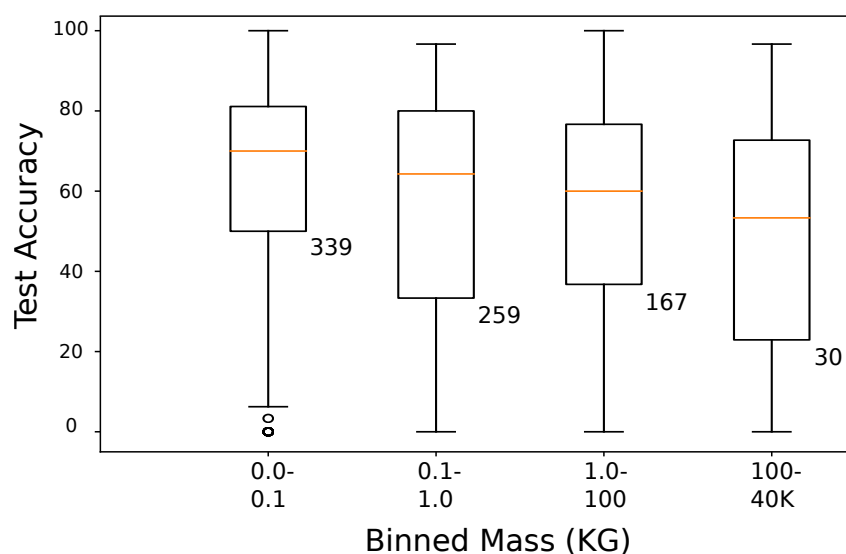


Figure 6.6: Top one public test set accuracy per class for (Szegedy, Ioffe, et al., 2016) for a subset of 795 classes of birds and mammals binned according to mass. The number of classes appears to the bottom right of each box.

explained by the reduced number of images for these species in the dataset.

Finally, in Figure 6.8 we examine the relationship between the number of images and the validation accuracy. The median number of training images per class for our entire training set is 41. For this experiment, we capped the maximum number of training images per class to 10, 20, 50, or all, and trained a separate Inception V3

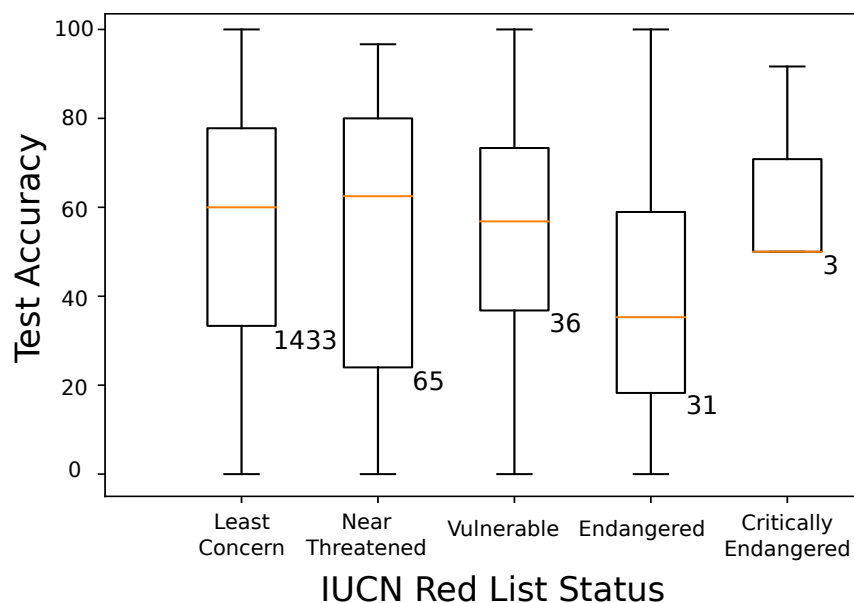


Figure 6.7: Top one public test set accuracy for (Szegedy, Ioffe, et al., 2016) for a subset of 1,568 species binned according to their IUCN Red List of Threatened Species status (Baillie, Hilton-Taylor, and Stuart, 2004). The number of classes appears to the bottom right of each box.

for each case. This corresponds to starting with 50,000 for the case of 10 images per class and then doubling the total amount of training data each time. For each species, we randomly selected the images up until the maximum amount. As noted in the main paper, more attention is needed to improve performance in the low data regime.

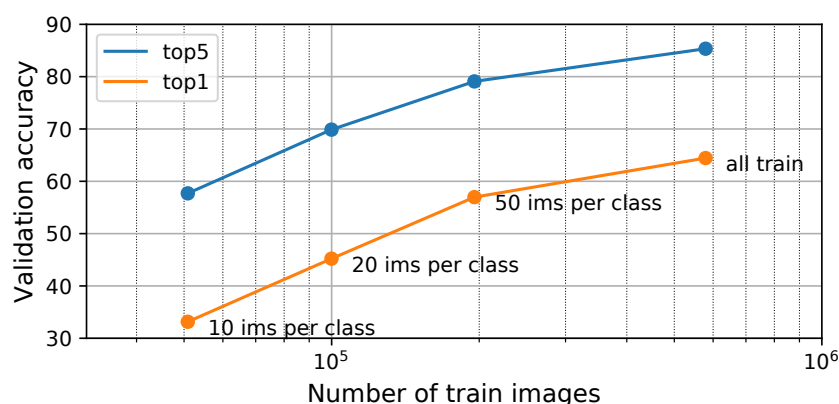


Figure 6.8: As the maximum number of training images per class increases, so does the accuracy. However, we observe diminishing returns as the number of images increases. Results are plotted on the validation set for the Inception V3 network (Szegedy, Vanhoucke, et al., 2016).

### iNat2017 Competition Results

From April to mid July 2017, we ran a public challenge on the machine learning competition platform Kaggle<sup>3</sup> using iNat2017. Similar to the classification tasks in (Russakovsky et al., 2015), we used the top five accuracy metric to rank competitors. We used this metric as some species can only be disambiguated with additional data provided by the observer, such as location or date. Additionally, in a small number of cases, multiple species may appear in the same image (e.g. a bee on a flower). Overall, there were 32 submissions and we display the final results for the top five teams in Table 6.5.

The top performing entry from *GMV* consisted of an ensemble of Inception V4 and Inception ResNet V2 networks Szegedy, Ioffe, et al., 2016. Each model was first initialized on the ImageNet-1K dataset and then finetuned with the iNat2017 training set along with 90% of the validation set, utilizing data augmentation at training time. The remaining 10% of the validation set was used for evaluation. To compensate for the imbalanced training data, the models were further fine-tuned on the 90% subset of the validation data that has a more balanced distribution. To address small object size in the dataset, inference was performed on  $560 \times 560$  resolution images using twelve crops per image at test time.

The additional training data amounts to 15% of the original training set, which, along with the ensembling, multiple test crops, and higher resolution, account for the improved 81.58% top 1 public accuracy compared to our best performing single model which achieved 68.53%.

Rank	Team name	Public Test		Private Test	
		Top1	Top5	Top1	Top5
1	GMV	<b>81.58</b>	<b>95.19</b>	<b>81.28</b>	<b>95.13</b>
2	Terry	77.18	93.60	76.76	93.50
3	Not hotdog	77.04	93.13	76.56	93.01
4	UncleCat	77.64	93.06	77.44	92.97
5	DLUT_VLG	76.75	93.04	76.19	92.96

Table 6.5: Final public challenge leaderboard results. ‘Rank’ indicates the final position of the team out of 32 competitors. These results are typically ensemble models, trained with higher input resolution, with the validation set as additional training data.

<sup>3</sup>[www.kaggle.com/c/inaturalist-challenge-at-fgvc-2017](http://www.kaggle.com/c/inaturalist-challenge-at-fgvc-2017)

## Detection Results

To characterize the detection difficulty of iNat2017, we adopt Faster-RCNN (Ren et al., 2017) for its state-of-the-art performance as an object detection setup (which jointly predicts object bounding boxes along with class labels). We use a TensorFlow (Abadi et al., 2016) implementation of Faster-RCNN with default hyperparameters (J. Huang et al., 2017). Each model is trained with 0.9 momentum and asynchronously optimized on 9 GPUs to expedite experiments. We use an Inception V3 network, initialized from ImageNet, as the backbone for our Faster-RCNN models. Finally, each input image is resized to have 600 pixels as the short edge while maintaining the aspect ratio.

As discussed in Section 6.4, we collected bounding boxes on 9 of the 13 super-classes, translating to a total of 2,854 classes with bounding boxes. In the following experiments, we only consider performance on this subset of classes. Additionally, we report performance on the validation set in place of the test set, and we only evaluate on images that contained a single instance. Images that contained only evidence of the species' presence and images that contained multiple instances were excluded. We evaluate the models using the detection metrics from COCO (T.-Y. Lin et al., 2014).

We first study the performance of fine-grained localization and classification by training the Faster-RCNN model on the 2,854 class subset. Figure 6.10 shows some sample detection results. Table 6.6 provides the break down in performance for each super-class, where super-class performance is computed by taking an average across all classes within the super-class. The precision-recall curves (again at the super-class level) for 0.5 IoU are displayed in Figure 6.9. Across all super-classes we achieve a comprehensive average precision (AP) of 43.5. Again the Reptilia super-class proved to be the most difficult, with an AP of 21.3 and an AUC of 0.315. At the other end of the spectrum, we achieved an AP of 49.4 for Insecta and an AUC of 0.677. Similar to the classification results, when viewed as a collection of datasets (one for each super-class), we see that iNat2017 exhibits highly variable detection difficulty, posing a challenge to researchers to build improved detectors that work across a broad group of fine-grained classes.

Next we explored the effect of label granularity on detection performance. We trained two more Faster-RCNN models, one trained to detect super classes rather fine-grained classes (so 9 classes in total), and another model trained with all labels pooled together, resulting in a generic object / not object detector. Table 6.7

shows the resulting AP scores for the three models when evaluated at different granularities. When evaluated on the coarser granularity, detectors trained on finer-grained categories have lower detection performance when compared with detectors trained at coarser labels. The performance of the 2,854-class detector is particularly poor on super-class recognition and object localization. This suggests that the Faster-RCNN algorithm has plenty of room for improvements on end-to-end fine-grained detection tasks.

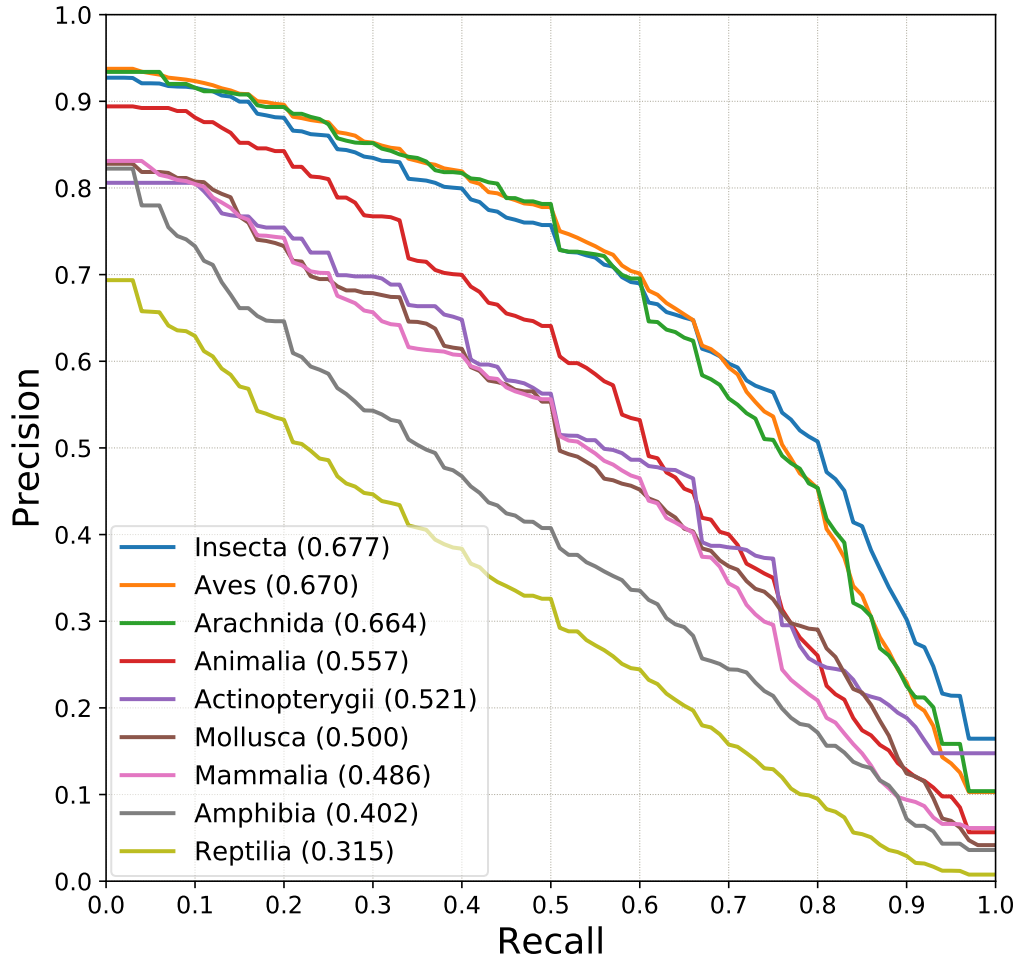


Figure 6.9: Precision-Recall curve with 0.5 IoU for each super-class, where the Area-Under-Curve (AUC) corresponds to  $AP^{50}$  in Table 6.6. Super-class performance is calculated by averaging across all fine-grained classes. We can see that building a detector that works well for all super-classes in iNat2017 will be a challenge.

### Additional Detection Results

In Table 6.8, we investigate detector performance for the 2,854-class model across different bounding box sizes using the size conventions of the COCO dataset (T.-Y. Lin et al., 2014). As expected, performance is directly correlated with size, where



	<b>AP</b>	<b>AP<sup>50</sup></b>	<b>AP<sup>75</sup></b>	<b>AR<sup>1</sup></b>	<b>AR<sup>10</sup></b>
Insecta	49.4	<b>67.7</b>	<b>59.3</b>	<b>64.5</b>	<b>64.9</b>
Aves	<b>49.5</b>	67.0	59.1	63.3	63.6
Reptilia	21.3	31.5	24.9	44.0	44.8
Mammalia	33.3	48.6	39.1	49.8	50.6
Amphibia	28.7	40.2	35.0	52.0	52.3
Mollusca	34.8	50.0	41.6	52.0	53.0
Animalia	35.6	55.7	40.8	48.3	50.5
Arachnida	43.9	66.4	49.6	57.3	58.6
Actinopterygii	35.0	52.1	41.6	49.1	49.6
Overall	43.5	60.2	51.8	59.3	59.8

Table 6.6: Super-class-level Average Precision (AP) and Average Recall (AR) for object detection, where AP, AP<sup>50</sup> and AP<sup>75</sup> denotes AP@[IoU=.50:.05:.95], AP@[IoU=.50] and AP@[IoU=.75] respectively; AR<sup>1</sup> and AR<sup>10</sup> denotes AR given 1 detection and 10 detections per image.

<b>Training</b>			<b>Evaluation</b>
	2854-class	9-super-class	1-generic
2854-class	43.5	55.6	63.7
9-super-class	-	65.8	76.7
1-generic	-	-	78.5

Table 6.7: Detection performance (AP@[IoU=.50:.05:.95]) with different training and evaluation class granularity. Using finer-grained class labels during training has a negative impact on coarser-grained super-class detection. This presents an opportunity for new detection algorithms that maintain precision at the fine-grained level.

smaller objects are more difficult to detect. However, examining Table 6.9, we can see that total number of these small instances is low for most super-classes.

## 6.6 Conclusions and Future Work

We present the iNat2017 dataset, in contrast to many existing computer vision datasets it is: (1) unbiased, in that it was collected by non-computer vision researchers for a well defined purpose; (2) more representative of real-world challenges than previous datasets; (3) represents a long-tail classification problem; and (4) is useful in conservation and field biology. The introduction of iNat2017 enables us to study two important questions in a real world setting: (1) do long-tailed datasets present intrinsic challenges; and (2) do our computer vision systems exhibit transfer learning from the well-represented categories to the least represented ones. While our baseline classification and detection results are encouraging, from

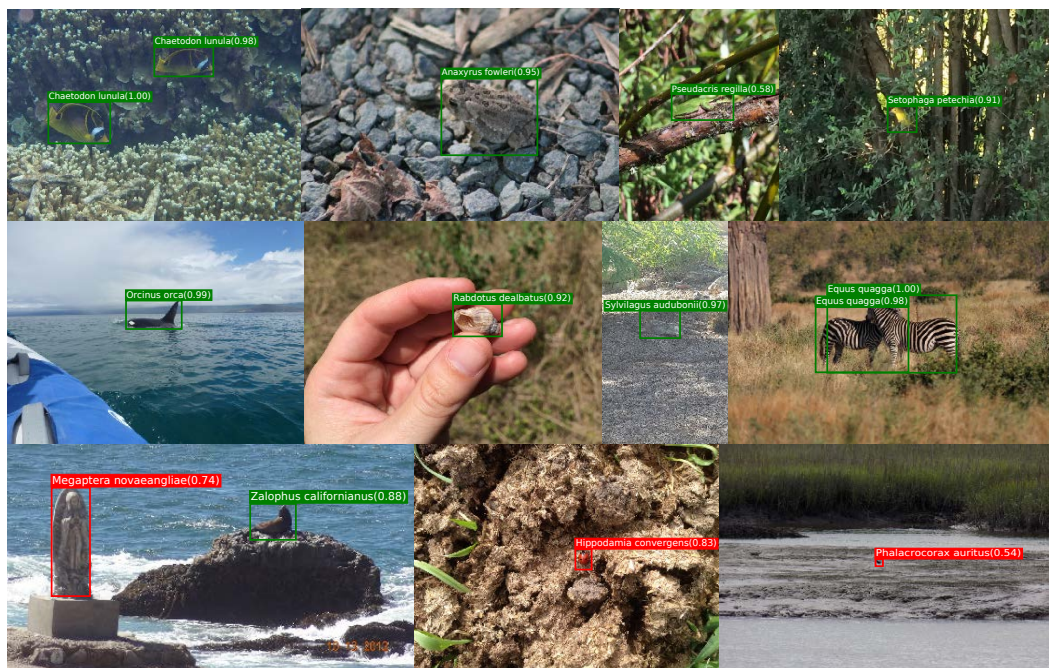


Figure 6.10: Sample detection results for the 2,854-class model that was evaluated across all validation images. Green boxes represent correct species level detections, while reds are mistakes. The bottom row depicts some failure cases. We see that small objects pose a challenge for classification, even when localized well.

our experiments we see that state-of-the-art computer vision models have room to improve when applied to large imbalanced datasets. Small, efficient models designed for mobile applications and embedded devices have even more room for improvement (Howard et al., 2017).

Unlike traditional, researcher-collected datasets, the iNat2017 dataset has the opportunity to grow with the iNaturalist community. Currently, every 1.7 hours another species passes the 20 unique observer threshold, making it available for inclusion in the dataset (already up to 12k as of November 2017, up from 5k when we started work on the dataset). Thus, the current challenges of the dataset (long tail with sparse data) will only become more relevant.

In the future we plan to investigate additional annotations such as sex and life stage attributes, habitat tags, and pixel level labels for the four super-classes that were challenging to annotate. We also plan to explore the “open-world problem” where the test set contains classes that were never seen during training. This direction would encourage new error measures that incorporate taxonomic rank (Mittal et al., 2012; Yan et al., 2015). Finally, we expect this dataset to be useful in studying how to teach fine-grained visual categories to humans (Singla et al., 2014; Johns,



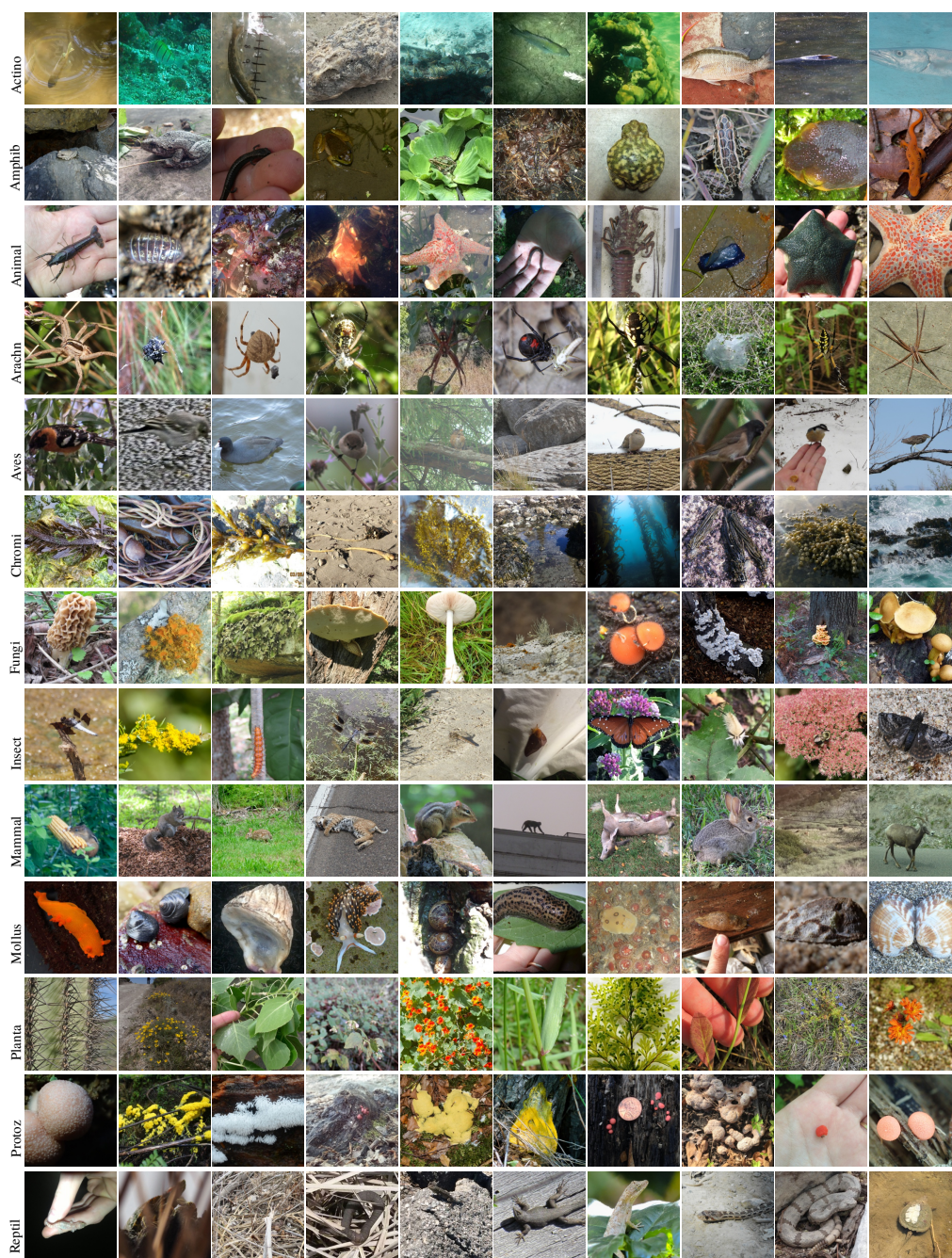


Figure 6.11: Example images from the training set. Each row displays randomly selected images from each of the 13 different super-classes. For ease of visualization we show the center crop of each image.

	<b>AP<sup>S</sup></b>	<b>AP<sup>M</sup></b>	<b>AP<sup>L</sup></b>	<b>AR<sup>S</sup></b>	<b>AR<sup>M</sup></b>	<b>AR<sup>L</sup></b>
Insecta	13.4	34.7	51.8	13.5	38.9	67.7
Aves	11.5	41.7	55.1	13.3	49.2	69.9
Reptilia	0.0	12.4	22.0	0.0	16.3	46.5
Mammalia	6.7	27.8	37.1	9.0	36.1	55.8
Amphibia	0.0	23.2	29.9	0.0	28.7	54.9
Mollusca	17.5	30.8	35.8	17.5	33.6	55.9
Animalia	24.0	22.7	37.1	26.7	28.2	52.0
Arachnida	16.2	32.9	46.5	16.2	38.5	61.6
Actinopterygii	5.0	16.3	36.1	5.0	17.9	51.1
Overall	11.0	34.7	46.7	12.5	40.7	63.7

Table 6.8: Super-class level Average Precision (AP) and Average Recall (AR) with respect to object sizes. S, M and, L denote small (area  $< 32^2$ ), medium ( $32^2 \leq \text{area} \leq 96^2$ ) and, large (area  $> 96^2$ ) objects. The AP for each super-class is calculated by averaging the results for all species belonging to it. Best and worst performance for each metric are marked by green and red, respectively.

	<b>Small</b>	<b>Medium</b>	<b>Large</b>
Insecta	445	2432	16429
Aves	2375	8898	16239
Reptilia	32	400	5426
Mammalia	280	1068	2751
Amphibia	20	253	2172
Mollusca	74	466	1709
Animalia	72	414	1404
Arachnida	12	152	909
Actinopterygii	32	144	634

Table 6.9: The number of super-class instances at each bounding box size in the validation set. While AP and AR is low for some super-classes at a particular size (see Table 6.8), the actual number of instances at that size may also be low.

Mac Aodha, and Brostow, 2015), and plan to experiment with models of human learning.

**Acknowledgments.** This work was supported by a Google Focused Research Award. We would like to thank: Scott Loarie and Ken-ichi Ueda from iNaturalist; Steve Branson, David Rolnick, Weijun Wang, and Nathan Frey for their help with the dataset; Wendy Kan and Maggie Demkin from Kaggle; the iNat2017 competitors, and the FGVC2017 workshop organizers. We also thank NVIDIA and Amazon Web Services for their donations.

## References

- Abadi, Martin et al. (2016). “Tensorflow: Large-scale machine learning on heterogeneous distributed systems”. In: *arXiv preprint arXiv:1603.04467*.
- Baillie, Jonathan, Craig Hilton-Taylor, and Simon N Stuart (2004). *2004 IUCN red list of threatened species: a global species assessment*. IUCN.
- Berg, Thomas et al. (2014). “Birdsnap: Large-scale fine-grained visual categorization of birds”. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, pp. 2019–2026.
- Cao, Qiong et al. (2017). “VGGFace2: A dataset for recognising faces across pose and age”. In: *arXiv preprint arXiv:1710.08092*.
- Cardinale, Bradley J et al. (2012). “Biodiversity loss and its impact on humanity”. In: *Nature*.
- Everingham, Mark et al. (2010). “The pascal visual object classes (voc) challenge”. In: *IJCV*.
- Fei-Fei, Li, Rob Fergus, and Pietro Perona (2007). “Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories”. In: *CVIU*.
- Gebru, Timnit et al. (2017). “Fine-grained car detection for visual census estimation”. In: *AAAI*.
- Guo, Yandong et al. (2016). “Ms-celeb-1m: A dataset and benchmark for large-scale face recognition”. In: *ECCV*.
- He, Kaiming et al. (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.
- Hou, Saihui, Yushan Feng, and Zilei Wang (2017). “VegFru: A Domain-Specific Dataset for Fine-grained Visual Categorization”. In: *ICCV*.
- Howard, Andrew G et al. (2017). “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861*.
- Hu, Jie, Li Shen, and Gang Sun (2017). “Squeeze-and-Excitation Networks”. In: *arXiv preprint arXiv:1709.01507*.
- Huang, Gary B. et al. (2007). *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. University of Massachusetts, Amherst.
- Huang, Jonathan et al. (2017). “Speed/accuracy trade-offs for modern convolutional object detectors”. In: *CVPR*.
- Johns, Edward, Oisín Mac Aodha, and Gabriel J Brostow (2015). “Becoming the expert-interactive multi-class machine teaching”. In: *CVPR*.

- Jones, Kate E et al. (2009). “PanTHERIA: a species-level database of life history, ecology, and geography of extant and recently extinct mammals”. In: *Ecology*.
- Khosla, Aditya et al. (2011). “Novel Dataset for Fine-Grained Image Categorization”. In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO.
- Krasin, I et al. (2016). “OpenImages: A public dataset for large-scale multi-label and multiclass image classification”. In: *Dataset available from <https://github.com/openimages>*.
- Krause, Jonathan, Benjamin Sapp, et al. (2016). “The unreasonable effectiveness of noisy data for fine-grained recognition”. In: *European Conference on Computer Vision*. Springer, pp. 301–320.
- Krause, Jonathan, Michael Stark, et al. (2013). “3d object representations for fine-grained categorization”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561.
- Kumar, Neeraj et al. (2012). “Leafsnap: A computer vision system for automatic plant species identification”. In: *Computer Vision–ECCV 2012*. Springer, pp. 502–516.
- Lin, Tsung-Yi et al. (2014). “Microsoft COCO: Common objects in context”. In: *ECCV*.
- Lin, Yen-Liang et al. (2014). “Jointly optimizing 3d model fitting and fine-grained classification”. In: *Computer Vision–ECCV 2014*. Springer, pp. 466–480.
- Lislevand, Terje, Jordi Figuerola, and Tamás Székely (2007). “Avian body sizes in relation to fecundity, mating system, display behavior, and resource sharing”. In: *Ecology*.
- Liu, Jiongxin et al. (2012). “Dog breed classification using part localization”. In: *Computer Vision–ECCV 2012*. Springer, pp. 172–185.
- Maji, Subhransu et al. (2013). “Fine-grained visual classification of aircraft”. In: *arXiv preprint arXiv:1306.5151*.
- Mittal, Arpit et al. (2012). “Taxonomic multi-class prediction and person layout using efficient structured ranking”. In: *ECCV*.
- Mora, Camilo et al. (2011). “How many species are there on Earth and in the ocean?” In: *PLoS biology* 9.8, e1001127.
- Nilsback, Maria-Elena and Andrew Zisserman (2006). “A visual vocabulary for flower classification”. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. IEEE, pp. 1447–1454.
- Parkhi, O. M. et al. (2012). “Cats and Dogs”. In: *CVPR*.
- Parkhi, Omkar M, Andrea Vedaldi, Andrew Zisserman, et al. (2015). “Deep Face Recognition.” In: *BMVC*.

- Ren, Shaoqing et al. (2017). “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *PAMI*.
- Rolnick, David et al. (2017). “Deep Learning is Robust to Massive Label Noise”. In: *arXiv preprint arXiv:1705.10694*.
- Russakovsky, Olga et al. (2015). “Imagenet large scale visual recognition challenge”. In: *International Journal of Computer Vision* 115.3, pp. 211–252.
- Schroff, Florian, Dmitry Kalenichenko, and James Philbin (2015). “Facenet: A unified embedding for face recognition and clustering”. In: *CVPR*.
- Singla, Adish et al. (2014). “Near-Optimally Teaching the Crowd to Classify.” In: *ICML*.
- Szegedy, Christian, Sergey Ioffe, et al. (2016). “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *arXiv preprint arXiv:1602.07261*.
- Szegedy, Christian, Wei Liu, et al. (2015). “Going deeper with convolutions”. In: *CVPR*.
- Szegedy, Christian, Vincent Vanhoucke, et al. (2016). “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.
- Taigman, Yaniv et al. (2014). “Deepface: Closing the gap to human-level performance in face verification”. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, pp. 1701–1708.
- Ueda, K (2017). “iNaturalist Research-grade Observations via GBIF.org.” In: URL: <https://doi.org/10.15468/ab3s5x>.
- Van Horn, Grant et al. (2015). “Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604. DOI: 10.1109/CVPR.2015.7298658.
- Vedaldi, Andrea et al. (2014). “Understanding objects in detail with fine-grained attributes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3622–3629.
- Wah, Catherine et al. (2011). “The caltech-ucsd birds-200-2011 dataset”. In:
- Wegner, Jan D et al. (2016). “Cataloging public objects using aerial and street-level images-urban trees”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6014–6023.
- Welinder, Peter et al. (2010). “Caltech-UCSD birds 200”. In:
- Wilber, Michael J et al. (2017). “BAM! The Behance Artistic Media Dataset for Recognition Beyond Photography”. In: *ICCV*.
- Xie, Saining et al. (2017). “Aggregated residual transformations for deep neural networks”. In: *CVPR*.

- Yan, Zhicheng et al. (2015). “HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition”. In: *ICCV*.
- Yang, Linjie et al. (2015). “A large-scale car dataset for fine-grained categorization and verification”. In: *CVPR*.
- Yu, Aron and Kristen Grauman (2014). “Fine-grained visual comparisons with local learning”. In: *CVPR*.
- Zhang, Xiao et al. (2017). “The iMaterialist Challenge 2017 Dataset”. In: *FGVC Workshop at CVPR*.



## REDUCING MEMORY & COMPUTATION DEMANDS FOR LARGE SCALE VISUAL CLASSIFICATION

Van Horn, Grant and Pietro Perona (2019). “Reducing Memory & Computation Demands for Large Scale Visual Classification”.

### **7.1 Abstract**

The computational and storage costs of state-of-the-art deep networks that are designed for large scale visual classification ( $>1K$  categories) is dominated by the fully-connected classification layers. This makes deployment problematic on mobile devices, where app download size and power efficient execution is critical. In this work we analyze different techniques aimed at reducing this bottleneck and present a new technique, Taxonomic Parameter Sharing, that utilizes a taxonomy to share parameters among the classes. Our experiments on the iNaturalist dataset show that a simple tactic of jointly training a standard fully connected layer along with a rank factorized layer can result in a 25x reduction in memory and computation in the classification layer without any loss in top-1 accuracy. The standard fully connected layer can be discarded at test time. For a task with 8k classes, this reduces the floating point memory requirements of the final layer from 64MB to 2.6MB when using a feature vector of size 2048. Our Taxonomic Parameter Sharing approach is competitive in the regime where reduced parameter count is important during both training and testing.

### **7.2 Introduction**

Deep convolutional neural networks (DCNN) have dramatically improved performance of computer vision systems. This includes decreases in error rates on academic benchmark datasets (Krizhevsky, Sutskever, and G. E. Hinton, 2012), fast and accurate retrieval performance on consumer devices (Howard et al., 2017), niche computer vision apps (Van Horn, Branson, et al., 2015), and rapidly improving self driving cars. A remaining obstacle to the ubiquitous adoption of DCNNs are the computational, energetic and memory costs of running the networks on portable wireless devices.

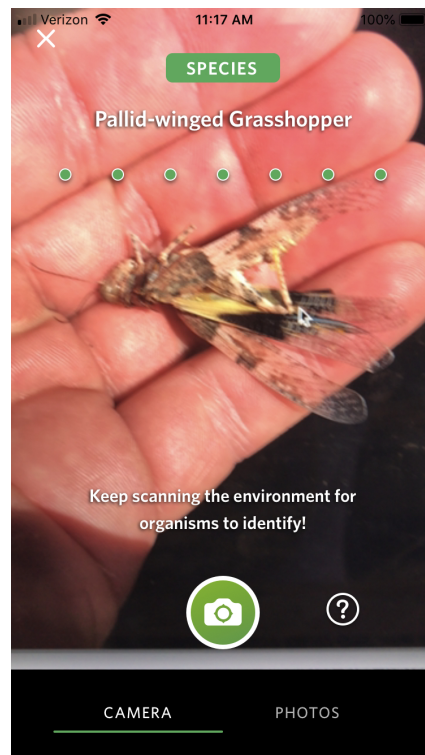


Figure 7.1: **Large Scale Visual Classification:** This work is motivated by a specific application problem: allow a user to point their phone’s camera at wildlife and classify it in real time without requiring a network connection. The model is trained with data collected by the citizen science website iNaturalist (<https://inaturalist.org>) and the number of species grows daily, totalling over 30k in January 2019. Important aspects of this application is the model size (which directly impacts the app download size), the execution time (which directly impacts user interaction), model efficiency (which directly impacts battery life) and the classification accuracy of the model (which directly impacts user satisfaction).

Two approaches have been proposed for reducing the computational and memory requirements of the network. The first focuses on reducing the precision of the network weights and activations by quantizing their values (i.e. representing numbers with fewer bits) (Asanovic and Morgan, 1991; Vanhoucke, Senior, and M. Z. Mao, 2011; Yunchao Gong et al., 2014; Courbariaux, Y. Bengio, and David, 2015; Han, H. Mao, and Dally, 2016; Rastegari et al., 2016). Besides reducing storage, if appropriate hardware is designed (Jouppi et al., 2017) the execution time and power can be reduced as well.

The second focuses on reducing the number of operations in the network. This is accomplished in four different ways. **1.** By designing efficient network architectures (Mamalet and Garcia, 2012; J. Jin, Dundar, and Culurciello, 2014; Szegedy

et al., 2016; Howard et al., 2017; Sandler et al., 2018; Zhang et al., 2018; X. Jin et al., 2018) (e.g. replacing  $7 \times 7$  convolution blocks with sequences of  $3 \times 3$  blocks). Similar to this are works that employ structured efficient linear layers (Yang et al., 2015; Cheng et al., 2015; Sindhvani, T. Sainath, and Kumar, 2015; Moczulski et al., 2015; Hoffer, Hubara, and Soudry, 2018) that allow for fast matrix multiplication with fewer parameters. Note that “convolutional networks” are themselves a means of parameter efficiency when compared to fully connected layers, locally connected features (Coates, Ng, and Lee, 2011) and tiled convolutional networks (Gregor and LeCun, 2010). **2.** Through network pruning (Hassibi and Stork, 1993; LeCun, Denker, and Solla, 1990; Han, Pool, et al., 2015; Guo, Yao, and Chen, 2016; Alvarez and Salzmann, 2016; Zhou, Alvarez, and Porikli, 2016; H. Li et al., 2017), where redundant weights (and therefore operations) are post-hoc removed from trained network. **3.** Through knowledge distillation (Buciluă, Caruana, and Niculescu-Mizil, 2006; Ba and Caruana, 2014; G. Hinton, Vinyals, and J. Dean, 2014) where a smaller network is trained on the logits or softmax output of a larger network or an ensemble of networks. **4.** Through filter factorization and decomposition techniques (Masana et al., 2017; Jaderberg, Vedaldi, and Zisserman, 2014; Mamalet and Garcia, 2012; Denton et al., 2014; Lebedev et al., 2014; Ba and Caruana, 2014) or rank restrictions (Xue, J. Li, and Yifan Gong, 2013; Denil et al., 2013; T. N. Sainath et al., 2013) to speed up a network and reduce memory usage.

Here we focus on the computational and storage costs of networks that classify thousands of categories, see Figure 7.1. This regime is often referred to as “Large-Scale Visual Classification” (LSVC). In LSVC computational and memory costs are dominated by the fully connected classification layer, which scales linearly with the number of classes. When the number of classes becomes sufficiently large, this final layer becomes the memory bottleneck of the network, and its multiply-add operations dominate computational costs. For example, consider a task with 8k classes and a backbone DCNN architecture that produces a 2k dimension feature vector. The fully connected layer for this setup has 16M parameters. If 32 bit floating point values are used, then this matrix consumes 64MB of memory, and takes 16M multiply-add operations to project the feature vector into class logits. Attempting to quantize the representation to 8 bits will only reduce the size by a factor 4, and will not reduce the number of multiply-add operations. Now consider the training regime where a batch size of 32, 64, or 128 is used; the memory demands balloon to over 8GB for a batch size of 128. We aim for dramatic cost reduction.

A straightforward tactic for reducing the cost is to factorize the fully connected layer into two lower rank matrices. In our experiments we explore the performance differences when we factorize as a post processing step, while training, and while fine-tuning. We also investigate the effects of jointly training a full sized layer along side a rank factorized layer. These techniques represent the simplest, easiest to implement tactics and we find that they can result in good performance. In addition to these strong baseline experiments, we propose a new architecture for the fully connected last layer of the network. This new architecture is composed of multiple fully connected layers whose outputs are parsed to navigate a taxonomy over the classes. We call this technique Taxonomic Parameter Sharing (TPS), and we demonstrate that it achieves competitive accuracy with much lower cost compared to the standard fully connected layer.

### 7.3 Related Work

#### Matrix Factorization

Our analysis of factorizing the final fully connected layer is related to the works of Sainath et al. (T. N. Sainath et al., 2013) and Denton et al. (Denton et al., 2014). Sainath et al. (T. N. Sainath et al., 2013) explored factorizing the final matrix to make **training** more efficient while preserving accuracy. We expand upon their work and focus specifically on test time efficiency by investigating factorized layers that are fine-tuned from or jointly trained with a non-factorized fully connected layer. Our experiments reveal that a 25x reduction in the final layer parameter count can be achieved without a loss in accuracy. Denton et al. (Denton et al., 2014) also focus on test time efficiency and present results on factorized fully connected layers (see Table 2 in (Denton et al., 2014)). We expand upon their work by investigating various techniques for training the factorized layers, provide a more thorough analysis of performance, and show that the parameters in the classification layer can be reduced by 25x, as opposed to their 8x findings.

#### Large-Scale Classification

In the realm of large scale classification (Deng, Dong, et al., 2009; Thomee et al., 2016; Krasin et al., 2017; Van Horn, Mac Aodha, et al., 2018) our work is related to methods that utilize a hierarchy to trade off concept specificity versus accuracy (Deng, Berg, et al., 2010; Deng, Krause, et al., 2012; Ordonez et al., 2013) and methods that use the hierarchy to learn experts on subsets of the classes (Yan et al., 2015; Ahmed, Baig, and Torresani, 2016). Most relevant are methods aimed

at reducing the computational bottleneck of the softmax layer of a neural network. These methods have been explored mainly along three directions.

The first direction is that of hierarchical models (Morin and Y. Bengio, 2005; Mikolov et al., 2013; Yan et al., 2015) (for SVM approaches see (Griffin and Perona, 2008; Marszałek and Schmid, 2008; Gao and Koller, 2011)). In these approaches a classifier is learned at each internal node of a taxonomic tree built on top of the class labels, which could be as simple as a two layer “course-to-fine” hierarchy. This increases the memory requirements of the model (the number of classes to classify has increased) but the computational requirements decreases to that of traversing to a leaf node. A similar approach is undertaken by (S. Bengio, Weston, and Grangier, 2010; Deng, Satheesh, et al., 2011; Liu et al., 2013) where the authors build a label tree over the classes, allowing for a reduction in both memory and computation demands. Our TPS approach is different in that instead of repeatedly dividing leaf nodes to learn a label tree we encode an existing tree structure. While at first this seems like a step backwards, the use of a semantic tree in our approach allows us to utilize additional training data not available to other algorithms and to provide interpretable taxonomic predictions. Further, our method achieves memory reduction in addition to computation reduction.

The second direction of research utilizes Locality Sensitive Hashing (Gionis, Indyk, Motwani, et al., 1999) to find the k-nearest rows of the weight matrix and then approximate the softmax output by using only the dot products between those k vectors and the feature vector. Vijayanarasimhan et al. (Vijayanarasimhan et al., 2015), building on the work of (Yagnik et al., 2011; T. Dean et al., 2013), introduce this hashing logic into both the training and inference executions of the model. The related works of (Mussmann and Ermon, 2016; Mussmann, Levy, and Ermon, 2017; Levy, Chan, and Ermon, 2018) have continued to explore this route, particularly for natural language processing. These methods can significantly decrease the computational cost of evaluating the softmax layer. However, the full weight matrix still needs to be made accessible. The hashing layers and random memory accesses that are necessary for these methods complicate GPU utilization.

The third direction of research (Krizhevsky and G. E. Hinton, 2011; Weston, S. Bengio, and Usunier, 2011) also utilizes hashing but do so in the context of k-nearest-neighbor search in an embedding space. These methods learn an embedding space representation (typically using a ranking loss), project the training data into this embedding space, and use k-nearest-neighbors via hashing to classify images

at inference time. The computation bottleneck is alleviated with these methods, but the memory requirements is increased due to storage of the embedded training images.

A similar theme with all of these works is that the reduction in computational complexity of evaluating the softmax layer requires increasing the memory requirements of the model or adding complex logic to the training or inference portions of the model, or both. We contribute a new technique that can both reduce the computation and memory requirements of the model, while still being simple.

#### 7.4 Taxonomic Parameter Sharing

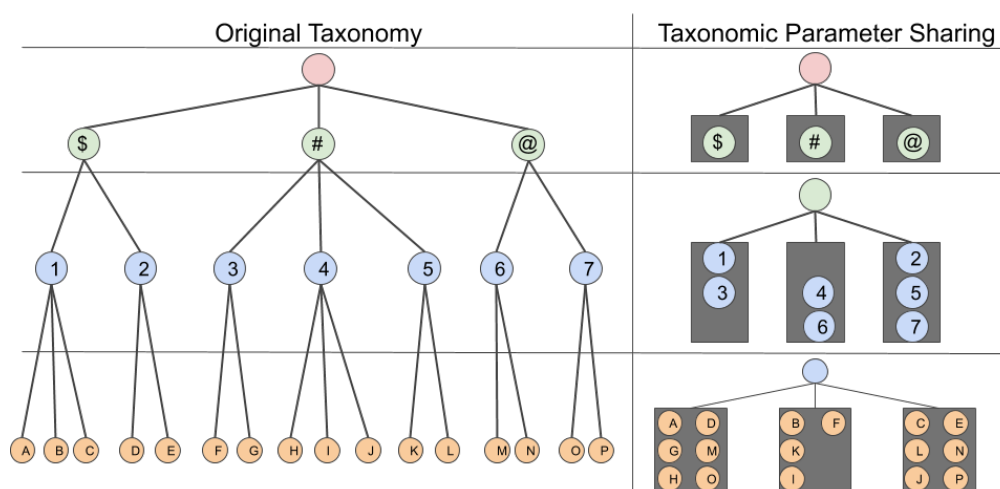


Figure 7.2: **Taxonomic Parameter Sharing:** In this visual example our method converts an original classification problem over 16 classes to 3 3-way classification problems by using a taxonomy over the original classes, a 1.7x reduction. At each level in the original taxonomy, we construct a new classification problem by binning the sibling nodes into distinct “buckets”. Nodes can be randomly assigned to “buckets”, or a distance metric and a more sophisticated assignment function can be used. Note that all sibling nodes from the original taxonomy are placed into separate “buckets” for the new classification problem. During training, all of the images from all of the nodes in a given “bucket” are combined together to train a given new class. This format make it trivial to include additional inner node training data from the original taxonomy. Each new classifier is trained jointly, receiving the same feature vector from the backbone DCNN (so 3 classification “heads” are trained in this example). At test time, all of the new classifiers are run, and a prediction on the original classes is obtained in the following way: (1) The red classifier is run, producing a prediction for either \$, #, or @. The green classifier is then run, and the most likely bucket that contains a child of the ancestor prediction is selected. Finally this process is repeated for the blue classifier, which selects a leaf node from the original taxonomy.

Our Taxonomic Parameter Sharing (TPS) method is a simple, greedy algorithm that utilizes a taxonomy over  $N$  classes to construct a collection of new classification problems  $C$ . The combined classification results from  $C$  can be used to predict the original  $N$  classes. Assume we have a taxonomy with  $L$  levels, where a node's level is the distance from it to the root node, and all leaf nodes are on the same level. Traditionally, a taxonomy is utilized by training a classifier at each inner node of the taxonomy to disambiguate that node's children. This increases the total number of classification tasks, but during test time we only need to traverse to a leaf node using the classification results from  $L$  classifiers. This can be far faster than evaluating one classifier over all leaf nodes. The motivation behind the TPS algorithm is to try to maintain the reduction in test time computation **and** reduce the total memory usage as well. We do this by grouping non-sibling nodes at the same level into new "super" classes. Therefore, we have exactly 1 classifier responsible for making predictions at each level. This is how the TPS algorithm reduces the memory demands while still requiring only  $L$  classifications to make a prediction.

Algorithm 2 provides an overview of the process of creating the new classification problems and Figure 7.2 provides a visual description. Given a taxonomy  $T$  our algorithm proceeds to process each level of the taxonomy independently. For each level  $l$ , we collect all the nodes at that level (i.e. all nodes at distance  $l$  from the root node) and first determine the largest number of siblings  $s_{\max}$  (i.e. the largest number of nodes at level  $l$  that share the same parent). We then construct a new classification problem with  $n_l$  classes where  $n_l \geq s_{\max}$ . We then assign all nodes at level  $l$  to one of the new classes. The algorithm is called Taxonomic Parameter Sharing because we are requiring multiple nodes at a given level in the taxonomy to share parameters. Next we provide two different methods for picking  $n_l$  and assigning the nodes to classes. We then describe how classification results on the new classes can be used to produce classes over the original  $N$  classes. And we finish this section with additional benefits of the proposed algorithm.

### Random Assignment

The simplest assignment strategy is to randomly assign each node at level  $l$  to one of the new classes  $n_l$ . Sibling nodes can be handled by grouping them together and sampling class assignments without replacement, so that it is guaranteed that no two siblings are assigned to the same class. Choosing the number of classes can be done greedily by taking  $n_l = s_{\max}$ . This produces the smallest classification problem for this level. Increasing  $n_l$  increases the number of classes that must be classified, but

**Algorithm 2** Taxonomic Parameter Sharing

---

```

1: input: taxonomy  $T$  with levels  $L$ 
2: for  $l \in L$  do
3:    $s_{\max} \leftarrow$  max sibling count at level  $l$ 
4:   Construct new classification problem  $C_l$  of size  $|C_l| \geq s_{\max}$ 
5:   Assign each node at level  $l$  to one of the new classes such that no sibling
     nodes are assigned to the same class.
6: end for
7: return  $C = \{C_l \forall l\}$  ▷ The new classification problems.

```

---

can also make the classification problem easier.

**Facility Location Assignment**

A more sophisticated assignment strategy is to determine the node assignment and number of classes  $n_l$  for level  $l$  jointly. This can be done by posing the problem as a facility location problem (Erlenkotter, 1978) and using a greedy approximation algorithm (Jain, Mahdian, and Saberi, 2002) to solve it. In this setup the cities are the nodes at level  $l$  and the facilities that are opened to service the cities correspond to the new classes. We can enforce that sibling nodes at level  $l$  cannot be assigned to the same facility. Facility location problems require a notion of distance between the cities and in our experiments we use the euclidean distance computed between averaged training feature vectors for each class (extracted from an ImageNet Inception-v3 model (Szegedy et al., 2016)). Alternatively, domain specific or application specific knowledge can be used to specify the distance of the nodes. Facility location problems also require a cost of opening a facility. This cost value is directly related to the final size of the new classification task, with a larger cost producing fewer opened facilities. We employed large cost values in our experiments so that the resulting number of classes is small.

**Classification**

We can convert the classification results on the new collection of classification tasks  $C$  for a test image to a classification of the original  $N$  classes using the taxonomy  $T$ . We start from the root node of  $T$  and choose the highest scoring class from  $n_1$ . Note that because all nodes at level  $l = 1$  are siblings, the highest scoring class from  $n_1$  corresponds to a unique node in the taxonomy, call this node  $t_1$ . We choose  $t_1$  as our prediction for level  $l = 1$ . For level  $l = 2$ , we examine the results of  $n_2$  and consider only those classes that contain children of  $t_1$ . Because all sibling nodes



are assigned to unique classes in  $n_2$ , we can simply choose the child corresponding with the highest scoring class from  $n_2$ . We repeat this process until we reach a leaf node, which will correspond to one of the original  $N$  classes. Note that conditional probabilities can be computed by multiplying the class probabilities as we traverse from the root node to a leaf node. Also, there is nothing preventing us from computing the conditional probability of all leaf nodes (no additional classifications need to be done), providing the full distribution across all of the original  $N$  classes.

### Inner Node Training Data

The Taxonomic Parameter Sharing algorithm makes it easy to include in the training set data that is not labeled at the species level, but rather at the genus or other level. These images, classified to level  $l$ , can simply be used to train all classifiers corresponding to levels  $\leq l$ . See Sec. 7.5 for details.

## 7.5 Experiments



Figure 7.3: **iNaturalist Images:** Example images from the iNaturalist 2018 competition dataset. Each column contains two different species from the same genus. These species pairs are often confused by the classifier. From left to right: *A. hetzi* and *A. chalcodes*, *P. thoas* and *P. rumiko*, *S. jello* and *S. barracuda*, *L. alleni* and *L. californicus*. Image credits from left to right, top to bottom: cullen, Roberto Gonzalez, Ian Banks, Francisco Farriols Sarabia, CK Kelly, Francisco Farriols Sarabia, Marisa Agarwal, mbalame99

### Dataset

We conduct experiments using an augmented version of the iNaturalist 2018 competition dataset <sup>1</sup>, see Figure 7.3. The iNaturalist 2018 dataset consists of 8,142 species, however, due to a taxonomy conflict with the taxonomy hosted on inaturalist.org, we removed 4 species from the dataset (classes 330, 5150, 119, 120) and we

<sup>1</sup>[https://github.com/visipedia/inat\\_comp](https://github.com/visipedia/inat_comp)

merged class 5185 with 5188 and class 6184 with 6185, resulting in 8,136 species and a taxonomy with complete ancestry paths (consisting of Kingdom, Phylum, Class, Order, Family, and Genus ancestors for all species), see Table 7.1 for the number of nodes and the maximum sibling counts at each taxonomic rank (we use the terms “rank” and “level” interchangeably). Our augmented dataset contained all of the iNaturalist 2018 dataset images (for the 8,136 species included), which are all identified to species.

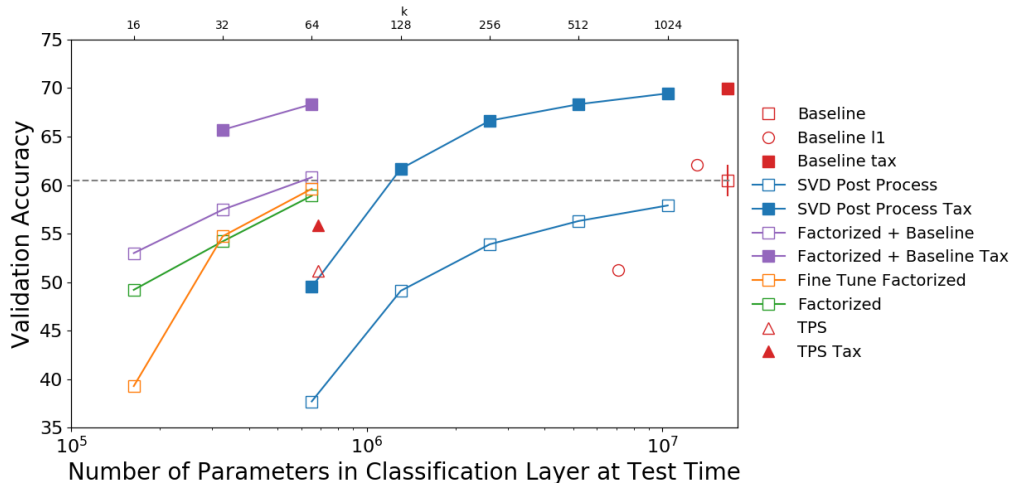
To explore the utility of training with non-leaf node data we augmented the dataset to include images identified to a courser node. These additional images were chosen by the following procedure: starting from genus nodes and continuing up to ancestors nodes, for each node  $n$  we sum the images in the descendants of  $n$  and attempt to include additional images identified to  $n$  until the total number of images is 1k. Note that when we are augmenting  $n$  we do not include images identified to descendants of  $n$ , we only include those images that the iNaturalist community identified at  $n$ . This procedure resulted in an additional 969,095 images added to the dataset, for a total of 1,406,529 training images. See Table 7.1 for a break down of how many additional images were included at each rank. Note that no additional images identified to species were included. We report accuracy metrics using the validation set from the iNaturalist 2018 dataset. All of the validation images are identified to species. In the following sections, performance numbers on the validation set refer to the percentage of images correctly identified when the model gets one guess.

	Kingdom	Phylum	Class	Order	Family	Genus	Species
# of Nodes	6	20	54	275	1114	4420	8136
Max Siblings	6	9	7	39	73	173	28
Images	215	680	2,006	18,993	162,806	784,395	437,434
Total Images	1,406,529	1,406,314	1,405,634	1,403,628	1,384,635	1,221,829	437,434

Table 7.1: **Taxonomy & Image Statistics:** Our iNaturalist taxonomy is composed of 8,136 species nodes, each with a Kingdom, Phylum, Class, Order, Family, and Genus ancestor. The non-taxonomic experiments use only the species training images. The taxonomic experiments use the species training images along with additional inner node data. The TPS models can use either just species data or can be trained with the additional inner node data as well. For the randomly assigned TPS models, the “Max Siblings” row provides the size of the respective classification problem at each level in the taxonomy. The “Total Images” row is a cumulative count of training data at a particular level along with the training data at lower levels.

### Backbone Architecture

We use an Inception-V3 (Szegedy et al., 2016) backbone architecture for all experiments. Image inputs are resized to 299x299 and basic image augmentations are employed. The model is trained using RMSProp with a batch size of 32, an initial learning rate of 0.0045 decayed by 0.94 every 4 epochs, batch normalization, and a small  $l_2$  regularization is applied to all weights. Training is monitored by plotting the training and validation performance and early stopping is employed. The output feature dimension is 2048. Unless otherwise stated, all experiments started from an ImageNet pretrained model.



**Figure 7.4: Validation Accuracy vs Test Time Parameter Count:** This plot summarizes our experiments on the iNaturalist 2018 dataset. The open symbols represent experiments that make use of only species-level labels, and **did not** make use of inner node data (Sec. 7.5). The filled symbols represent experiments that **did** make use of both species-level and inner node data (Sec. 7.5).  $\square$  symbols represent  $l_2$  regularized fully connect or factorized experiments.  $\circ$  symbols represent  $l_1$  regularized fully connected experiments.  $\triangle$  symbols represent our Taxonomic Parameter Sharing (TPS) model (Sec. 7.5). Red experiments are baselines of the respective models. Blue experiments take a trained model from a red experiment and factorize the fully connected classification layer using SVD (Sec. 7.5 and Sec. 7.5). Green experiments train factorized matrices of the form  $2048 \times k$  and  $k \times 8136$  from scratch (Sec. 7.5). Orange experiments fine-tune factorized matrices of the form  $2048 \times k$  and  $k \times 8136$ , starting from a fully trained baseline model (Sec. 7.5). Purple experiments jointly train standard fully connected layers and factorized matrices (i.e. multiple classification heads)(Sec. 7.5 and Sec. 7.5). Jointly training a standard fully connected layer along with a factorized version (purple curves) provides the best parameter reduction to accuracy loss. Jointly training a rank 64 factorized matrix reduces the parameters in the classification layer by 25x (at test time) and results in no loss in accuracy.

## Non-Taxonomic Models

The models used in this section do not make use of a taxonomy nor any of the additional training available on the inner nodes.

### Baseline

Our baseline method simply trains a fully connected layer of equal size to the number of species, resulting in a matrix of size  $2048 \times 8136$  with 16,662,528 parameters. Note that we do not include the bias in the parameter count for any of the models. This “off the shelf” model achieves a top-1 accuracy of  $60.5 \pm 1.7$ . We will use these values as baselines for the rest of the methods. This model represents an “out-of-the-box” solution.

### $l_1$ Regularization

In these experiments we took our baseline model and regularized the last fully connected layer using an  $l_1$  penalty. This penalty encourages weights to be 0, so it is the optimizer that is tasked with increasing the sparsity of the last layer. We experimented with varying regularization strengths of  $4^{-4}$ ,  $4^{-5}$ , and  $4^{-6}$ , resulting in top-1 accuracy scores of 8.82, 51.21, and 62.12. If we clip all weight values whose absolute value is less than  $1^{-7}$ , then these models would produce sparse final fully connected layers with 16,199,031, 7,096,449, and 13,149,137 respectively. We can see that  $4^{-4}$  was too strong of a regularization and resulted in a model that could not converge, while a smaller  $l_1$  regularization like  $4^{-6}$  resulted in a high performing model, but with only a (hypothetical) factor of 1.26x savings in memory.

### SVD

In these experiments we took our baseline model (fully trained) and factorized the last matrix using SVD, producing three lower rank matrices  $U\Sigma V^T$ . We then classified each validation image using the factorized model. We experimented with the following lower rank values: 64, 128, 256, 512, and 1024. Figure 7.4 plots the accuracy and parameter counts of these models. We can see that small rank values produce desired reduction in the number of parameters, but the accuracy takes a significant hit, with a rank 64 factorization resulting in a top-1 accuracy of 37.71.

## Matrix Factorization

In these experiments, as opposed to doing an SVD post processing operation on a trained matrix, we train a factorized fully connected layer, composed of two lower rank matrices of size  $2048 \times k$  and  $k \times 8136$ . We experimented with the following rank values  $k \in \{16, 32, 64\}$ . Figure 7.4 plots the accuracy and parameter counts of these models (blue curve, non-filled squares). Note that these models were trained from an ImageNet model. We can see that this is a simple method that results in a significant decrease in parameters while maintaining reasonable accuracy. This result was similarly mentioned by Denton et al. (Denton et al., 2014)), see Section 5 of their work.

## Matrix Factorization Fine-tuning

In these experiments we take the trained baseline model and replace the large fully connected layer with randomly initialized factorized matrices (of the form  $2048 \times k$  and  $k \times 8136$ ). We then fine-tune **only** the factorized matrices, leaving the backbone network untouched. We experimented with the following rank values  $k \in \{16, 32, 64\}$ . Figure 7.4 plots the accuracy and parameter counts of these models (orange curve, non-filled squares). We can see that we can effectively recover the performance of the baseline model with much fewer parameters (651,776 vs 16,662,528 for rank 64). However, performance does drop for even lower rank values (39.3 top-1 accuracy for rank 16).

## Baseline + Matrix Factorization Joint Training

In these experiments we jointly train a baseline model (i.e. fully connected layer of size  $2048 \times 8136$ ) along with a factorized fully connected layer, composed of two lower rank matrices of size  $2048 \times k$  and  $k \times 8136$ . To be specific, the backbone produces a feature vector of size 2048 which is fed into two different “classification heads”, one that is a standard fully connected layer and one that is a factorized version of a fully connected layer. Losses are computed for both outputs and are added equally. We experimented with the following rank values  $k \in \{16, 32, 64\}$ . Figure 7.4 plots the accuracy and parameter counts of these models (purple curve, non-filled squares). We can see that this method recovers a high performing factorized layer for low rank factorizations, performing as well or better than the fine-tuned factorizations. We can also see that the factorized performance for low rank values ( $k = 32$  and  $k = 16$ ) still results in good performance (57.5

and 53 top-1 accuracy, respectively). A rank  $k = 16$  factorization results in a 102x savings in weights and computations .

### **Taxonomic Models**

The models in this section make use of the taxonomy for training, and some make use of the taxonomy for classification. It will be clear from the context which models make use of the additional training data at the inner nodes during training.

#### **Baseline**

Our baseline model utilizes the taxonomy by training a separate fully connected layer for each of the 7 taxonomic ranks, see Table 7.1 for details on the number of nodes at each rank. During training, this model uses  $2048 \times 14,025 \approx 28\text{M}$  parameters for the classification layer. Once this model is trained, we keep only the species classifier and remove the other classifiers for testing. Using the additional inner node training data, our taxonomic baseline species classifier achieves a top-1 accuracy of 70.0. Note that while at test time this model has the same number of parameters in the classification layer as the non-taxonomic baseline, during training this taxonomic baseline requires 1.7x more parameters.

#### **SVD**

In this experiment we decompose the fully connected layer of the species classifier from the taxonomic model using SVD. We experimented with the following rank values  $k \in \{64, 128, 256, 512, 1024\}$ . Figure 7.4 plots the accuracy and parameter counts of these models (blue curve, filled squares). Similar to the non-taxonomic baseline, we can see that accuracy is well maintained for large rank values (i.e.  $k = 1024$ ) but falls off for smaller rank values (i.e.  $k = 64$ ) that produce the desired large decreases in parameters.

#### **Baseline + Matrix Factorization Joint Training**

In this experiment we trained a taxonomic baseline model (i.e. 7 fully connected layers for each of the ranks) plus an additional factorized species model. We then classified the validation images use the factorized species classifier. We experimented with the following rank values  $k \in \{32, 64\}$ . Figure 7.4 plots the accuracy and parameter counts of these models (purple curve, filled squares). Similar to the non-taxonomic experiments, we can see the benefit of doing this joint training,

where we recover essentially the same performance of the baseline model but use far fewer parameters.

### Taxonomic Parameter Sharing

In these experiments we analyze the performance of TPS models. Using a random assignment method to the fewest number of classes possible (see Table 7.1 for the max siblings at each level), the TPS model achieves a top-1 accuracy of 51.2 using a total of 686,080 parameters (a reduction in parameters by 24.3x compared to the baseline). This is better accuracy than the 47.75 accuracy of the trained matrix factorization of rank  $k = 64$  method (with 651,776 parameters), with both methods starting from an ImageNet pretrained model. We achieved similar performance using the facility location assignment algorithm to bin the nodes into the new classification problems. Unlike the previous methods, incorporating additional inner node training data requires no additional parameters during training. The additional training data increased the performance of the randomly assigned TPS model to 55.9.

The TPS model makes predictions in a hierarchical manner, meaning that it is affected by mistakes made at ancestor classifiers. To put the TPS model's performance into context, we can compare it to the baseline (full rank) taxonomic model that makes its predictions via hierarchical predictions (i.e. predict Kingdom before predicting Phylum, *etc.*) as opposed to only species classifications. The baseline taxonomic model trained **without** additional data achieves a top-1 accuracy of 54.2 using hierarchical predictions. The baseline taxonomic model trained **with** additional data achieves a top-1 accuracy of 59.8 using hierarchical predictions. Using these values as upper limits of performance, we can see that the TPS models are within 94% and 93% of these limits when trained without and with the additional data, respectively, but use 24.3x fewer parameters.

### Observations

We note a few observations from these experiments. First, it is advantageous to be able to train a standard fully connected layer because the minimum found by this over-parameterized layer is better than what can be discovered by a more parameter constrained layer. Second, either fine-tuning a factorized layer from a model trained with a standard fully connected layer, or (more preferably) jointly training a factorized layer with the standard fully connected layer, enables the factorized layer to achieve a performance comparable to the fully connected layer

but with drastically fewer parameters. Being able to jointly train the factorized layer means that only one training pass needs to be done. Third, if the number of classes is too large to train a standard fully connected layer, or training a standard fully connected layer is simply too slow, then the TPS algorithm can achieve as good or better performance than a factorized method. The TPS algorithm gets the additional benefit of being able to include additional inner node training data at no additional parameter expense.

## 7.6 Conclusion

In this work we analyzed several different methods for reducing the computation and memory requirements of the classification layer, including the novel Taxonomic Parameter Sharing algorithm. We used the large-scale iNaturalist 2018 competition dataset to conduct the experiments and arrived at several interesting findings. Likely the most interesting and useful for general practitioners is the fact that one can jointly train a factorized classification matrix with the regular fully connected layer to produce a test time model that maintains the same accuracy, yet uses 25x fewer parameters. This is a big savings when considering the usage of DCNN in mobile applications, where users are wary of large download sizes, and power consumption is a top concern.

For future work we plan on exploring an even larger class space, for the natural world this will take us to over one million species. Towards this end it will be beneficial to take into account the work on dynamic routing in networks (McGill and Perona, 2017). Having one layer responsible for hundreds of thousands or millions of classes seems undesirable. Exploring how networks can organize their knowledge into clusters and dynamically access that knowledge based on the input seems like a more manageable way forward.

## References

- Ahmed, Karim, Mohammad Haris Baig, and Lorenzo Torresani (2016). “Network of experts for large-scale image categorization”. In: *European Conference on Computer Vision*. Springer, pp. 516–532.
- Alvarez, Jose M and Mathieu Salzmann (2016). “Learning the number of neurons in deep networks”. In: *Advances in Neural Information Processing Systems*, pp. 2270–2278.
- Asanovic, Krste and Nelson Morgan (1991). *Experimental determination of precision requirements for back-propagation training of artificial neural networks*. International Computer Science Institute.



- Ba, Jimmy and Rich Caruana (2014). “Do deep nets really need to be deep?” In: *Advances in neural information processing systems*, pp. 2654–2662.
- Bengio, Samy, Jason Weston, and David Grangier (2010). “Label embedding trees for large multi-class tasks”. In: *Advances in Neural Information Processing Systems*, pp. 163–171.
- Buciluă, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil (2006). “Model compression”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 535–541.
- Cheng, Yu et al. (2015). “An exploration of parameter redundancy in deep networks with circulant projections”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2857–2865.
- Coates, Adam, Andrew Ng, and Honglak Lee (2011). “An analysis of single-layer networks in unsupervised feature learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223.
- Courbariaux, Matthieu, Yoshua Bengio, and Jean-Pierre David (2015). “Binaryconnect: Training deep neural networks with binary weights during propagations”. In: *Advances in neural information processing systems*, pp. 3123–3131.
- Dean, Thomas et al. (2013). “Fast, accurate detection of 100,000 object classes on a single machine”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1814–1821.
- Deng, Jia, Alexander C Berg, et al. (2010). “What does classifying more than 10,000 image categories tell us?” In: *European conference on computer vision*. Springer, pp. 71–84.
- Deng, Jia, Wei Dong, et al. (2009). “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pp. 248–255.
- Deng, Jia, Jonathan Krause, et al. (2012). “Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 3450–3457.
- Deng, Jia, Sanjeev Satheesh, et al. (2011). “Fast and balanced: Efficient label tree learning for large scale object recognition”. In: *Advances in Neural Information Processing Systems*, pp. 567–575.
- Denil, Misha et al. (2013). “Predicting parameters in deep learning”. In: *Advances in neural information processing systems*, pp. 2148–2156.
- Denton, Emily L et al. (2014). “Exploiting linear structure within convolutional networks for efficient evaluation”. In: *Advances in neural information processing systems*, pp. 1269–1277.
- Erlenkotter, Donald (1978). “A dual-based procedure for uncapacitated facility location”. In: *Operations Research* 26.6, pp. 992–1009.

- Gao, Tianshi and Daphne Koller (2011). “Discriminative learning of relaxed hierarchy for large-scale visual recognition”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, pp. 2072–2079.
- Gionis, Aristides, Piotr Indyk, Rajeev Motwani, et al. (1999). “Similarity search in high dimensions via hashing”. In: *Vldb*. Vol. 99. 6, pp. 518–529.
- Gong, Yunchao et al. (2014). “Compressing deep convolutional networks using vector quantization”. In: *arXiv preprint arXiv:1412.6115*.
- Gregor, Karo and Yann LeCun (2010). “Emergence of complex-like cells in a temporal product network with local receptive fields”. In: *arXiv preprint arXiv:1006.0448*.
- Griffin, Gregory and Pietro Perona (2008). “Learning and using taxonomies for fast visual categorization”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, pp. 1–8.
- Guo, Yiwen, Anbang Yao, and Yurong Chen (2016). “Dynamic network surgery for efficient dnns”. In: *Advances In Neural Information Processing Systems*, pp. 1379–1387.
- Han, Song, Huizi Mao, and William J Dally (2016). “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding”. In: *ICLR*.
- Han, Song, Jeff Pool, et al. (2015). “Learning both weights and connections for efficient neural network”. In: *Advances in neural information processing systems*, pp. 1135–1143.
- Hassibi, Babak and David G Stork (1993). “Second order derivatives for network pruning: Optimal brain surgeon”. In: *Advances in neural information processing systems*, pp. 164–171.
- Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (2014). “Distilling the knowledge in a neural network”. In: *NIPS Deep Learning Workshop*.
- Hoffer, Elad, Itay Hubara, and Daniel Soudry (2018). “Fix your classifier: the marginal value of training the last weight layer”. In: *arXiv preprint arXiv:1801.04540*.
- Howard, Andrew G et al. (2017). “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861*.
- Jaderberg, Max, Andrea Vedaldi, and Andrew Zisserman (2014). “Speeding up convolutional neural networks with low rank expansions”. In: *arXiv preprint arXiv:1405.3866*.
- Jain, Kamal, Mohammad Mahdian, and Amin Saberi (2002). “A new greedy approach for facility location problems”. In: *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*. ACM, pp. 731–740.
- Jin, Jonghoon, Aysegul Dundar, and Eugenio Culurciello (2014). “Flattened convolutional neural networks for feedforward acceleration”. In: *arXiv preprint arXiv:1412.5474*.

- Jin, Xiaojie et al. (2018). “WSNet: Compact and Efficient Networks Through Weight Sampling”. In: *International Conference on Machine Learning*, pp. 2357–2366.
- Jouppi, Norman P et al. (2017). “In-datacenter performance analysis of a tensor processing unit”. In: *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*. IEEE, pp. 1–12.
- Krasin, Ivan et al. (2017). “OpenImages: A public dataset for large-scale multi-label and multi-class image classification.” In:
- Krizhevsky, Alex and Geoffrey E Hinton (2011). “Using very deep autoencoders for content-based image retrieval.” In: *ESANN*.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks.” In: *NIPS*.
- Lebedev, Vadim et al. (2014). “Speeding-up convolutional neural networks using fine-tuned cp-decomposition”. In: *arXiv preprint arXiv:1412.6553*.
- LeCun, Yann, John S Denker, and Sara A Solla (1990). “Optimal brain damage”. In: *Advances in neural information processing systems*, pp. 598–605.
- Levy, Daniel, Danlu Chan, and Stefano Ermon (2018). “LSH Softmax: Sub-Linear Learning and Inference of the Softmax Layer in Deep Architectures”. In:
- Li, Hao et al. (2017). “Pruning Filters for Efficient ConvNets”. In: *International Conference on Learning Representations*.
- Liu, Baoyuan et al. (2013). “Probabilistic label trees for efficient large scale image classification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 843–850.
- Mamalet, Franck and Christophe Garcia (2012). “Simplifying convnets for fast learning”. In: *International Conference on Artificial Neural Networks*. Springer, pp. 58–65.
- Marszałek, Marcin and Cordelia Schmid (2008). “Constructing category hierarchies for visual recognition”. In: *European conference on computer vision*. Springer, pp. 479–491.
- Masana, Marc et al. (2017). “Domain-Adaptive Deep Network Compression”. In: *The IEEE International Conference on Computer Vision (ICCV)*.
- McGill, Mason and Pietro Perona (2017). “Deciding how to decide: Dynamic routing in artificial neural networks”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 2363–2372.
- Mikolov, Tomas et al. (2013). “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*, pp. 3111–3119.
- Moczulski, Marcin et al. (2015). “ACDC: A structured efficient linear layer”. In: *arXiv preprint arXiv:1511.05946*.

- Morin, Frederic and Yoshua Bengio (2005). “Hierarchical probabilistic neural network language model.” In: *Aistats*. Vol. 5. Citeseer, pp. 246–252.
- Mussmann, Stephen and Stefano Ermon (2016). “Learning and inference via maximum inner product search”. In: *International Conference on Machine Learning*, pp. 2587–2596.
- Mussmann, Stephen, Daniel Levy, and Stefano Ermon (2017). “Fast amortized inference and learning in log-linear models with randomly perturbed nearest neighbor search”. In: *arXiv preprint arXiv:1707.03372*.
- Ordonez, Vicente et al. (2013). “From large scale image categorization to entry-level categories”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2768–2775.
- Rastegari, Mohammad et al. (2016). “Xnor-net: Imagenet classification using binary convolutional neural networks”. In: *European Conference on Computer Vision*. Springer, pp. 525–542.
- Sainath, Tara N et al. (2013). “Low-rank matrix factorization for deep neural network training with high-dimensional output targets”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 6655–6659.
- Sandler, Mark et al. (2018). “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520.
- Sindhwani, Vikas, Tara Sainath, and Sanjiv Kumar (2015). “Structured transforms for small-footprint deep learning”. In: *Advances in Neural Information Processing Systems*, pp. 3088–3096.
- Szegedy, Christian et al. (2016). “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.
- Thomee, Bart et al. (2016). “YFCC100M: The new data in multimedia research”. In: *Communications of the ACM* 59.2, pp. 64–73.
- Van Horn, Grant, Steve Branson, et al. (2015). “Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604. DOI: 10.1109/CVPR.2015.7298658.
- Van Horn, Grant, Oisin Mac Aodha, et al. (2018). “The inaturalist species classification and detection dataset”. In: *Computer Vision and Pattern Recognition (CVPR)*.
- Vanhoucke, Vincent, Andrew Senior, and Mark Z Mao (2011). “Improving the speed of neural networks on CPUs”. In: *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*. Vol. 1. Citeseer, p. 4.

- Vijayanarasimhan, Sudheendra et al. (2015). “Deep networks with large output spaces”. In: *Workshop for International Conference on Learning Representations*.
- Weston, Jason, Samy Bengio, and Nicolas Usunier (2011). “Wsabie: Scaling up to large vocabulary image annotation”. In: *IJCAI*. Vol. 11, pp. 2764–2770.
- Xue, Jian, Jinyu Li, and Yifan Gong (2013). “Restructuring of deep neural network acoustic models with singular value decomposition.” In: *Interspeech*, pp. 2365–2369.
- Yagnik, Jay et al. (2011). “The power of comparative reasoning”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, pp. 2431–2438.
- Yan, Zhicheng et al. (2015). “HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition”. In: *ICCV*.
- Yang, Zichao et al. (2015). “Deep fried convnets”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1476–1483.
- Zhang, Xiangyu et al. (2018). “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Zhou, Hao, Jose M Alvarez, and Fatih Porikli (2016). “Less is more: Towards compact cnns”. In: *European Conference on Computer Vision*. Springer, pp. 662–677.

*Chapter 8***CONCLUSIONS AND FUTURE DIRECTIONS**

In this thesis, motivated by the desire to build Visipedia, I, along with my coauthors, have contributed work aimed at improving our ability to collect computer vision datasets. Using iNaturalist and the Cornell Lab of Ornithology as case studies, we have explored how to interact with motivated enthusiasts, experts, and paid crowdworkers to collect and combine the necessary data to train computer vision models to answer visual questions. While the current applications help users around the world, there is still plenty of work to be done before we can achieve the vision of Visipedia laid out by Perona (Perona, 2010).

The Merlin and iNaturalist apps are image classification apps. They process the whole image, or a region selected by the user, and return a list of candidate species. This is the behavior expected by the user, but it is far from the behavior envisioned by Perona. The image does not become interactive and therefore does not allow the user to readily answer additional visual questions past, “What species is this?” I would argue it is not computer vision models (Krizhevsky, Sutskever, and Hinton, 2012; Ren et al., 2017; He et al., 2017) or efficient annotation tools (Branson, Van Horn, and Perona, 2017) that are missing, rather it is collecting the right type of data and conveniently rendering the information for the user. Essentially, we are still missing the two important interfaces that Visipedia must provide: an interface that allows experts to share their knowledge, and an interface that allows users to answer visual questions.

The expert interface must allow experts to easily contribute their knowledge. I would imagine that this type of interface would let experts browse a taxonomy of annotation types (imagine a taxonomy of objects and options for annotating different parts of those objects), giving them the ability to add additional nodes and options as needed, and encouraging them to annotate at the finest possible level. Annotations could be simple keypoints, boxes, lines, segmentations, or any one of the standard drawing tools now available on mapping interfaces or image editing interfaces. A more complicated, but more powerful, annotation interface would let the expert mark the correspondences with a 3D model of the object (in addition to allowing them to create the 3D models). This would allow structure and constraints to be

enforced in a vision model, and would also allow future annotations to be efficiently propagated (e.g. interpolate between two existing parts to annotate a third part that occurs between them). Interesting academic questions that arise here include: (1) how to select which images to ask the expert to annotate under the constraints of time, expert cost, and expected gain; (2) when to ask the expert to refine the taxonomy of annotations or 3D models due to ambiguity; and (3) how to propagate information down the taxonomy when new subtrees are added.

The question-answering interface is more devious than it seems at first. In Perona's vision (Perona, 2010), there are many automata that can annotate an image. How should we decide which automata to use when processing an image from a user? Given a collection of automata's output, how can we combine their information and render it on the image? I believe a taxonomy will help us here too. Rather than an image being analyzed once, I think it should be analyzed repeatedly based on the interactions of the user. If an image contains a hummingbird eating the nectar of a flower, rather than immediately covering the image with tens or hundreds of clickable regions, the user should be able to provide their intention by clicking on the bird or the flower. Then the component regions of that object should be rendered. This process should continue until the user has clicked on the hyperlink for a component region (taking them to Wikipedia). Interesting academic questions that arise here include: (1) how to manage a taxonomy of automata capable of producing component regions for an image (i.e. mitigate duplicates or conflicts); (2) how to efficiently traverse the taxonomy of automata when processing an image; and (3) how to assist the user when their target region of the image never becomes clickable or the rendered components are inaccurate.

The third academic question from the previous paragraph is relevant to the current versions of the Merlin and iNaturalist apps. Often, the list of results returned by the classifiers contains the correct species, however it is the user's job to sift through the example images to identify the match. This process could be drastically improved by designing better human-machine interfaces. The human visual system is very powerful, and can often augment the capabilities of the vision classifier, yet it is currently ignored. Returning to a twenty questions style interface would help users resolve ambiguous situations or continue their search in the wake of failed automata results. Taking the work from Branson et al. (Branson, Van Horn, Wah, et al., 2014) and updating it for use with convolutional networks and taxonomies would be a useful step forward.

## References

- Branson, Steve, Grant Van Horn, and Pietro Perona (2017). “Lean Crowdsourcing: Combining Humans and Machines in an Online System”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7474–7483. DOI: 10.1109/CVPR.2017.647.
- Branson, Steve, Grant Van Horn, Catherine Wah, et al. (2014). “The ignorant led by the blind: A hybrid human–machine vision system for fine-grained categorization”. In: *International Journal of Computer Vision* 108.1-2, pp. 3–29.
- He, Kaiming et al. (2017). “Mask r-cnn”. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, pp. 2980–2988.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks.” In: *NIPS*.
- Perona, Pietro (2010). “Vision of a Visipedia”. In: *Proceedings of the IEEE* 98.8, pp. 1526–1534.
- Ren, Shaoqing et al. (2017). “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *PAMI*.